

## N O T I C E

THIS DOCUMENT HAS BEEN REPRODUCED FROM  
MICROFICHE. ALTHOUGH IT IS RECOGNIZED THAT  
CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED  
IN THE INTEREST OF MAKING AVAILABLE AS MUCH  
INFORMATION AS POSSIBLE



National Aeronautics and  
Space Administration

DEC 15 1980

JSC-17024

Lyndon B. Johnson Space Center  
Houston, Texas 77058

NASA CR-161014

DETECTION AND MAPPING (DAM) PACKAGE

Volume 4b: Software System Manual (part 2)

(NASA-CR-161014) DETECTION AND MAPPING  
(DAM) PACKAGE. VOLUME 4B: SOFTWARE SYSTEM  
MANUAL, PART 2 Final Report (Lockheed  
Engineering and Management) 721 p  
HC A99/MF A01

N81-26750

Unclas  
CSCL 09B G3/61 30069

Edward H. Schlosser  
Lockheed Engineering & Managmeent Services Co., Inc.  
1830 NASA Road 1  
Houston, Texas 77058

NAS 9-15800

September 1980  
Final Report for Period January - September 1980

Prepared for  
LYNDON B. JOHNSON SPACE CENTER  
Houston, Texas 77058



LEMSCD-15616

1. Report No.		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle <b>DETECTION AND MAPPING (DAM) PACKAGE</b>  <b>Volume 4b: Software System Manual (part 2)</b>				5. Report Date <b>September 1980</b>	
				6. Performing Organization Code	
7. Author(s) <b>Edward H. Schlosser</b>				8. Performing Organization Report No. <b>LENSCO-15616</b>	
9. Performing Organization Name and Address <b>Lockheed Engineering &amp; Management Services Co., Inc.</b> <b>1830 NASA Road 1</b> <b>Houston, Texas 77058</b>				10. Work Unit No.	
				11. Contract or Grant No. <b>NAS 9-15800</b>	
12. Sponsoring Agency Name and Address <b>O. G. Smith</b> <b>Earth Observation Division</b> <b>Lyndon B. Johnson Space Center, Houston, Texas 77058</b>				13. Type of Report and Period Covered <b>Final, Jan.-Sept. 1980</b>	
				14. Sponsoring Agency Code	
15. Supplementary Notes Software available from: <b>COSMIC</b> <b>University of Georgia</b> <b>112 Barrow Hall</b> <b>Athens, Georgia 30602</b>					
16. Abstract  The DAM package is an integrated set of manual procedures, computer programs, and graphic devices designed for efficient production of precisely registered and formatted maps from digital Landsat multi-spectral scanner (MSS) data. The software can be readily implemented on any Univac 1100 series computer with standard peripheral equipment. This version of the software includes pre-defined spectral limits for use in classifying and mapping surface water for Landsat-1, Landsat-2, and Landsat-3. Tape formats supported include "X", "A", and "PM".					
17. Key Words (Suggested by Author(s))  <b>Computer programs, Earth resources, Earth satellites, land use, Landsat satellites, mapping, remote sensors, surface water, thematic mapping</b>				18. Distribution Statement	
19. Security Classif. (of this report)  <b>Unclassified</b>		20. Security Classif. (of this page)  <b>Unclassified</b>		21. No. of Pages	
				22. Price*	

\*For sale by the National Technical Information Service, Springfield, Virginia 22161

PREFACE

Multispectral scanners onboard NASA unmanned Landsat satellites provide an ideal source of current data for Earth resources applications. The Detection and Mapping (DAM) package was originally developed at the Johnson Space Center for rapid conversion of the Landsat digital data into hydrographic maps matching standard topographic quadrangle series. Recent improvements in both the manual procedures and computer programs within the DAM package make it easier to use, faster, and more general purpose.

Documentation and software for the DAM package are available to all public and private agencies, in accordance with the NASA policy of encouraging maximum use of remote sensing technology.

Published documentation, in which this is volume 4b, is comprised of the following volumes:

Volume 1: General Procedure

Volume 2: Software User Manual (in two parts)

Volume 3: Control Network Establishment

Volume 4: Software System Manual (in two parts)

These volumes supersede the previous documentation published in 1973. Software releases prior to version 7602 cannot be used with the current documentation.

Volume 4b contains software listings and documentation which have not been published prior to version 8009.

PRECEDING PAGE BLANK NOT FILMED



1 8009

DETECTION AND MAPPING PACKAGE  
.....

SYSTEM DESIGN  
-----

E H SCHLOSSER

PROGRAMMING  
-----

D A BECK  
H L BROWN  
J C CRISP  
H G EPPLER  
C A HELMKE  
H A HOLLEY  
T R KELL  
R E NARVESON  
J C POOLEY  
E H SCHLOSSER  
H A TOMPKINS

FUNDING  
-----

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION  
LYNDON B JOHNSON SPACE CENTER  
HOUSTON. TEXAS 77058

CONTRACTOR  
-----

LOCKHEED ENGINEERING AND MANAGEMENT SERVICES COMPANY  
HOUSTON. TEXAS 77058

PRECEDING PAGE BLANK NOT FILMED

8ASG.CP LISTPRINT..F/32/TRK/512  
8BRKPT PRINTS/LISTPRINT  
8HDD.P XXX DAM PACKAGE APPENDICES (VERSION 8009)  
8PRT.F DAM.  
8PRT.S DAM.SYS-TITLE  
8PRT.S DAM.SYS-LIST  
8HDD.P XXX DAM PACKAGE APPENDIX A: GENERAL DOCUMENTATION  
8ADD DAM.APPENDIX-A  
8HDD.P XXX DAM PACKAGE APPENDIX B: EXEC COMMAND DOCUMENTATION  
8ADD DAM.APPENDIX-B  
8HDD.P XXX DAM PACKAGE APPENDIX C: PROGRAM USER DOCUMENTATION  
8ADD DAM.APPENDIX-C  
8HDD.P XXX DAM PACKAGE APPENDIX D: COMMAND USER DOCUMENTATION  
8ADD DAM.APPENDIX-D  
8HDD.P XXX DAM PACKAGE APPENDIX E: MACRO COMMAND DOCUMENTATION  
8ADD DAM.APPENDIX-E  
8HDD.P XXX DAM PACKAGE APPENDIX F: SAMPLE CONTROL NETWORKS  
8ADD DAM.APPENDIX-F  
8HDD.P XXX DAM PACKAGE APPENDIX G: SPECTRAL LIMITS/TRANSFORMS  
8ADD DAM.APPENDIX-G  
8HDD.P XXX DAM PACKAGE APPENDIX H: SAMPLE RUNSTREAMS  
8ADD DAM.APPENDIX-H  
8HDD.P XXX DAM PACKAGE APPENDIX I: REVISIONS AND NEWS  
8ADD DAM.APPENDIX-I  
8HDD.P XXX DAM PACKAGE APPENDIX J: DEFAULT COMMANDS  
8ADD DAM.APPENDIX-J  
8HDD.P XXX DAM PACKAGE APPENDIX K: SYSTEM IMPLEMENTATION  
8ADD DAM.APPENDIX-K  
8HDD.P XXX DAM PACKAGE APPENDIX L: MAIN PROGRAMS/ROUTINES  
8ADD DAM.APPENDIX-L  
8HDD.P XXX DAM PACKAGE APPENDIX M: COMMAND ROUTINES  
8ADD DAM.APPENDIX-M  
8HDD.P XXX DAM PACKAGE APPENDIX N: UTILITY ROUTINES  
8ADD DAM.APPENDIX-N  
8HDD.P XXX DAM PACKAGE APPENDIX O: COORDINATE TRANSFORMATIONS  
8ADD DAM.APPENDIX-O  
8HDD.P XXX DAM PACKAGE APPENDIX P: EXECUTIVE REQUESTS  
8ADD DAM.APPENDIX-P  
8HDD.P XXX DAM PACKAGE APPENDIX Q: MACROS  
8ADD DAM.APPENDIX-Q  
8HDD.P XXX DAM PACKAGE APPENDIX R: CHAR/BYTE/STRING ROUTINES  
8ADD DAM.APPENDIX-R  
8HDD.P XXX DAM PACKAGE APPENDIX S: SORT ROUTINES  
8ADD DAM.APPENDIX-S  
8HDD.P XXX DAM PACKAGE APPENDIX Y: INTERNAL CODE/TESTING  
8ADD DAM.APPENDIX-Y  
8HDD.P XXX DAM PACKAGE APPENDIX Z: FILE DIRECTORY  
8PRT.TL DAM.  
8BRKPT PRINTS  
8FREE LISTPRINT.  
8PRT.F LISTPRINT.  
8SYM LISTPRINT.

(This volume contains Appendices N thru S)

**DAM PACKAGE APPENDIX A  
GENERAL DOCUMENTATION**

**APPENDIX-A  
001**

<b>SPRT.SC DAM.</b>	<b>APPENDIX-A . (0009)..A-1</b>
<b>SPRT.SC DAM.EXP----</b>	<b>.</b>
<b>SPRT.SC DAM.EXP-DAM</b>	<b>. DAM PACKAGE.....A-3</b>
<b>SPRT.SC DAM.EXP----</b>	<b>. ....</b>
<b>SPRT.SC DAM.EXP-LANDSAT</b>	<b>. LANDSAT.....A-5</b>
<b>SPRT.SC DAM.EXP-RULES</b>	<b>. RULES.....A-6.</b>
<b>SHORE:</b>	<b>. A-7</b>
<b>SPRT.SC DAM.EXP-FILES</b>	<b>. FILES.....A-8</b>
<b>SPRT.SC DAM.EXP-CARD</b>	<b>. CARD CODES.....A-9</b>
<b>SPRT.SC DAM.EXP-COORDINA</b>	<b>. COORDINATES.....A-10</b>
<b>SPRT.SC DAM.EXP-COMPUTER</b>	<b>. COMPUTER RUNS.....A-11</b>
<b>SPRT.SC DAM.EXP-LOCAL</b>	<b>. LOCAL STANDARDS.....A-12</b>

**DAM PACKAGE APPENDIX B  
EXEC COMMAND DOCUMENTATION**

**APPENDIX-B  
001**

<b>BPRT.SC DAM.</b>	<b>APPENDIX-B . (0000)..B-1</b>
<b>BPRT.SC DAM.EXP----</b>	<b>.</b>
<b>BPRT.SC DAM.EXP-EXEC</b>	<b>. SUMMARY.....B-3</b>
<b>BPRT.SC DAM.EXP-SADD</b>	<b>. SADD.....B-4</b>
<b>BPRT.SC DAM.EXP-SASO</b>	<b>. SASO.....B-5</b>
<b>BPRT.SC DAM.EXP-SCOPY</b>	<b>. SCOPY.....B-6</b>
<b>BPRT.SC DAM.EXP-SDATA/CHECKOUT</b>	<b>. SDATA/CHECKOUT.....B-7</b>
<b>BPRT.SC DAM.EXP-SED</b>	<b>. SED.....B-8</b>
<b>BPRT.SC DAM.EXP-SEND</b>	<b>. SEND.....B-9</b>
<b>BPRT.SC DAM.EXP-SEOF</b>	<b>. SEOF.....B-10</b>
<b>BPRT.SC DAM.EXP-SFIN</b>	<b>. SFIN.....B-11</b>
<b>BPRT.SC DAM.EXP-SFREE</b>	<b>. SFREE.....B-12</b>
<b>BPRT.SC DAM.EXP----</b>	<b>.</b>
<b>BPRT.SC DAM.EXP-SLOCATE</b>	<b>. SLOCATE.....B-14</b>
<b>BPRT.SC DAM.EXP----</b>	<b>.</b>
<b>BPRT.SC DAM.EXP----</b>	<b>.</b>
<b>BPRT.SC DAM.EXP-SREWIND</b>	<b>. SREWIND.....B-17</b>
<b>BPRT.SC DAM.EXP-SRUN</b>	<b>. SRUN.....B-18</b>
<b>BPRT.SC DAM.EXP-SUSE</b>	<b>. SUSE.....B-19</b>
<b>BPRT.SC DAM.EXP-SUSHAP</b>	<b>. SUSHAP.....B-20</b>
<b>BPRT.SC DAM.EXP-SXQT</b>	<b>. SXQT.....B-21</b>
<b>BPRT.SC DAM.EXP----</b>	<b>.</b>

DAM PACKAGE APPENDIX C  
PROGRAM USER DOCUMENTATION

APPENDIX-C  
001

SPRT.SC DAM.	APPENDIX-C . (0005) ..C-1
SPRT.SC DAM.EXP----	.
SPRT.SC DAM.EXP-PROGRAMS	. SUMMARY.....C-3
SPRT.SC DAM.EXP-ERTS-DUP	. ERTS-DUP.....C-4
SPRT.SC DAM.EXP-ERTSIDC	. ERTSIDC.....C-5
SPRT.SC DAM.EXP-PICTAB	. PICTAB.....C-6.
SHORE:	. C-7
SPRT.SC DAM.EXP-CONTROL	. CONTROL.....C-8.
SHORE:	. C-9
SPRT.SC DAM.EXP-CLASSIFY	. CLASSIFY.....C-10.
SHORE:	. C-11
SPRT.SC DAM.EXP-PRTDET	. PRTDET.....C-12.
SHORE:	. C-13
SPRT.SC DAM.EXP-PRTCLASS	. PRTCLASS.....C-14.
SHORE:	. C-15
SPRT.SC DAM.EXP-PLTCLASS	. PLTCLASS.....C-16
SPRT.SC DAM.EXP----	. .....
SPRT.SC DAM.EXP-STATUS	. STATUS.....C-18

DAM PACKAGE APPENDIX D  
COMMAND USER DOCUMENTATION

APPENDIX-D  
001

SPRT.SC DAM.	APPENDIX-D . (0009)..D-1.	
SHORE:		D-2
SPRT.SC DAM.EXP-COMMANDS	SUMMARY.....	D-3.
SHORE:		D-4.
SHORE:		D-5
SPRT.SC DAM.EXP----	.....	(FUTURE)
SPRT.SC DAM.EXP-ADJ	ADJUST.....	D-7
SPRT.SC DAM.EXP-ALI	ALIGN.....	D-8
SPRT.SC DAM.EXP----	.....	
SPRT.SC DAM.EXP-ATT	ATTITUDE.....	D-10
SPRT.SC DAM.EXP-CEN	CENTER.....	D-11
SPRT.SC DAM.EXP-CHA	CHANNEL.....	D-12
SPRT.SC DAM.EXP-CLE	CLEAR.....	D-13
SPRT.SC DAM.EXP-COL	COLOR.....	D-14
SPRT.SC DAM.EXP-COP	COPIES.....	D-15
SPRT.SC DAM.EXP-COU	COUNT.....	D-16
SPRT.SC DAM.EXP-CRO	CROSSTAB.....	D-17
SPRT.SC DAM.EXP----	.....	
SPRT.SC DAM.EXP-DEN	DENSITY.....	D-19
SPRT.SC DAM.EXP-OET	DETECT.....	D-20
SPRT.SC DAM.EXP-DIA	DIAGRAM.....	D-21
SPRT.SC DAM.EXP-DIS	DISPLAY.....	D-22
SPRT.SC DAM.EXP-EXI	EXIT.....	D-23
SPRT.SC DAM.EXP-EXP	EXPLAIN.....	D-24
SPRT.SC DAM.EXP-FAC	FACTOR.....	D-25
SPRT.SC DAM.EXP-FI	FI..(ENDIF).....	D-26
SPRT.SC DAM.EXP-GEO	GEOMETRY.....	D-27
SPRT.SC DAM.EXP----	.....	
SPRT.SC DAM.EXP-HEA	HEADING.....	D-29
SPRT.SC DAM.EXP----	HISTOGRAM.....	D-30
SPRT.SC DAM.EXP-IF	IF.....	D-31
SPRT.SC DAM.EXP-INT	INTENSITY.....	D-32
SPRT.SC DAM.EXP-LIN	LINEAR.....	D-33
SPRT.SC DAM.EXP-LIS	LIST.....	D-34
SPRT.SC DAM.EXP-MAP	MAP.....	D-35
SPRT.SC DAM.EXP-MER	MERIDIAN.....	D-36
SPRT.SC DAM.EXP----	.....	
SPRT.SC DAM.EXP-NAME	NAME.....	D-38
SPRT.SC DAM.EXP-NEW	NEWS.....	D-39
SPRT.SC DAM.EXP-NEX	NEXT.....	D-40
SPRT.SC DAM.EXP-OFF	OFF.....	D-41
SPRT.SC DAM.EXP-ON	ON.....	D-42
SPRT.SC DAM.EXP-ORI	ORIGIN.....	D-43
SPRT.SC DAM.EXP----	.....	
SPRT.SC DAM.EXP-PAGE	PAGE.....	D-45
SPRT.SC DAM.EXP----	.....	D-46
SPRT.SC DAM.EXP-PEE	PEEK.....	D-47
SPRT.SC DAM.EXP-PIC	PICTURE.....	D-48
SPRT.SC DAM.EXP----	PLOTTER.....	D-49
SPRT.SC DAM.EXP-POI	POINT.....	D-50
SPRT.SC DAM.EXP-POK	POKE.....	D-51
SPRT.SC DAM.EXP-POL	POLAR.....	D-52
SPRT.SC DAM.EXP-PRI	PRINTER.....	D-53
SPRT.SC DAM.EXP-PRO	PROFILE.....	D-54
SPRT.SC DAM.EXP----	.....	
SPRT.SC DAM.EXP-RAD	RADIANCE.....	D-56

SET TABS 8 27 6 98

(FUTURE)

ALTITUDE (FUTURE)

CRT (FUTURE)

(FUTURE)

HISTOGRAM (FUTURE)

MODEL (FUTURE)

(REPLACE WITH IF...FI)

(FUTURE)

PARTITION (FUTURE)

(FUTURE)

DAN PACKAGE APPENDIX D  
COMMAND USER DOCUMENTATION

APPENDIX-D  
002

SPRT.SC DAN.EXP-RAN	. RANK.....D-57
SPRT.SC DAN.EXP-REN	. RENUMBER.....D-58
SPRT.SC DAN.EXP-RES	. RESAMPLING.....D-59
SPRT.SC DAN.EXP-ROT	. ROTATE.....D-60
SPRT.SC DAN.EXP----	. .....D-61
SPRT.SC DAN.EXP-SCA	. SCALE.....D-62
SPRT.SC DAN.EXP-SCE	. SCENE.....D-63
SPRT.SC DAN.EXP----	. .....D-64
SPRT.SC DAN.EXP-SHA	. SHARPENING.....D-65
SPRT.SC DAN.EXP-SIZ	. SIZE.....D-66
SPRT.SC DAN.EXP----	. .....D-67
SPRT.SC DAN.EXP-SPA	. SPACING.....D-68
SPRT.SC DAN.EXP----	. SPHEROID.....D-69
SPRT.SC DAN.EXP-SYM	. SYMBOLS.....D-70
SPRT.SC DAN.EXP-TAB	. TABULATE.....D-71
SPRT.SC DAN.EXP----	. .....D-72
SPRT.SC DAN.EXP-TIC	. TICK.....D-73
SPRT.SC DAN.EXP-TIM	. TIME.....D-74
SPRT.SC DAN.EXP-TIN	. TINT.....D-75
SPRT.SC DAN.EXP-TOL	. TOLERANCE.....D-76
SPRT.SC DAN.EXP-TOT	. TOTAL.....D-77
SPRT.SC DAN.EXP----	. .....D-78
SPRT.SC DAN.EXP-WIN	. WINDOW.....D-79
SPRT.SC DAN.EXP-ZON	. ZONE.....D-80

SATURATION (FUTURE)

SCRIPT (FUTURE)

SKEW (FUTURE)

SPHEROID (FUTURE)

TERMINAL (FUTURE)

(FUTURE)

DAN PACKAGE APPENDIX E  
MACRO COMMAND DOCUMENTATION

APPENDIX-E  
001

SET TABS 3 20 6 30

```

SPRT.SC          DAN.    APPENDIX-E . (0000).....E-1
SPRT.SC DAN.EXP----
SPRT.SC DAN.EXP-MACRO . USING MACRO COMMANDS.....E-3
SPRT.SC DAN.EXP-MACDEF . DEFINING MACRO COMMANDS...E-4
SPRT.SC          DAN.PIC-ANO-RAD ....(PICTAB).....E-5
SPRT.SC          DAN.CLA-CIR-20 ....(CLASSIFY)....E-6
SPRT.SC          DAN.PIC-CIR-20 ....(PICTAB).....E-7.
SHORE:           E-8
SPRT.SC          DAN.PRO-CIR-20 ....(PRTOET).....E-9
SPRT.SC          DAN.CLA-CONFIRM ....(CLASSIFY)....E-10
SPRT.SC          DAN.CON-CONFIRM ....(CONTROL)....E-11
SPRT.SC          DAN.PIC-CONFIRM ....(PICTAB).....E-12
SPRT.SC          DAN.PRC-CONFIRM ....(PRCLASS)....E-13
SPRT.SC          DAN.CLA-HELP ....(CLASSIFY).....E-14
SPRT.SC          DAN.CON-HELP ....(CONTROL).....E-15
SPRT.SC          DAN.PIC-HELP ....(PICTAB).....E-16
SPRT.SC          DAN.PRC-HELP ....(PRCLASS).....E-17
SPRT.SC DAN.EXP----
SPRT.SC          DAN.    JSC-PRINT .....E-19
SPRT.SC DAN.EXP----
SPRT.SC          DAN.PIC-LIST-ALL ... (PICTAB).....E-21
SPRT.SC          DAN.PIC-LIST-5 ....(PICTAB).....E-22
SPRT.SC          DAN.    MAC4SPEC-1-2 .....E-23
SPRT.SC DAN.EXP----
SPRT.SC          DAN.    NED-PRINT .....E-25
SPRT.SC DAN.EXP----
SPRT.SC          DAN.PRC-Q1SHAP .....E-27
SPRT.SC          DAN.PRC-Q1SSPEC .....E-28
SPRT.SC          DAN.    SED-PRINT .....E-29
SPRT.SC DAN.EXP----
SPRT.SC          DAN.PIC-STAR-KM .....E-31
SPRT.SC          DAN.    STAR-SCAN .....E-32
SPRT.SC          DAN.    MES-PRINT .....E-33
SPRT.SC DAN.EXP----
SPRT.SC          DAN.    56X56H-L0 .....E-35
SPRT.SC          DAN.    56X56H-L0-R2 .....E-36
SPRT.SC          DAN.    56X56H-L0-R3 .....E-37
SPRT.SC          DAN.    56X56H-L0-R4 .....E-38
SPRT.SC          DAN.    56X56H-L0-R5 .....E-39
SPRT.SC DAN.EXP----
SPRT.SC          DAN.    56X56H-L0 .....E-41
SPRT.SC          DAN.    56X56H-L0-R2 .....E-42
SPRT.SC          DAN.    56X56H-L0-R3 .....E-43
SPRT.SC          DA.    56X56H-L0-R4 .....E-44

```



DAM PACKAGE APPENDIX F  
SAMPLE CONTROL NETWORKS

APPENDIX-F  
001

SPRT.SC DAM.APPENDIX-F . (0000).....F-1  
SPRT.SC  
SPRT.SC DAM.1027-10244 .....F-3  
SPRT.SC DAM.1027-10244/UTM .....F-4  
SPRT.SC DAM.1073-10244-3 ..(STRIP 3)....F-5  
SPRT.SC DAM.1092-10305 .....F-6  
SPRT.SC DAM.1092-10305-3 ..(STRIP 3)....F-7  
SPRT.SC  
SPRT.SC DAM.1132-10512 .....F-9  
SPRT.SC DAM.1191-19381 .....F-10  
SPRT.SC DAM.1205-19494 .....F-11  
SPRT.SC DAM.1205-10010 .....F-12  
SPRT.SC DAM.1209-10261 .....F-13  
SPRT.SC DAM.1302-15551 .....F-14  
SPRT.SC DAM.1407-19381 .....F-15  
SPRT.SC DAM.1420-10303 .....F-16  
SPRT.SC DAM.1420-10305 .....F-17  
SPRT.SC DAM.1704-10231 .....F-18  
SPRT.SC DAM.1768-20391 .....F-19  
SPRT.SC  
SPRT.SC DAM.21654-17513 .. ('PM') .....F-21  
SPRT.SC  
SPRT.SC DAM.30130-10032 .. ('X') .....F-23  
SPRT.SC

DAM.EXP----

DAM.EXP----

DAM.EXP----

DAM.EXP----

DAM.EXP----

DAM PACKAGE APPENDIX 0  
SPECTRAL LIMITS/TRANSFORMS

APPENDIX-0  
001

SPRT.SC DAM.APPENDIX-0 . (0000)..0-1.  
SHORE: . 0-2  
SPRT.SC DAM.WATER-LIN .....0-3  
SPRT.SC DAM.WATER-LIN/1 .....0-4  
SPRT.SC DAM.WATER-LIN/1-ANDERSON .....0-5  
SPRT.SC DAM.WATER-LIN/1-4IN-NARROW .....0-6  
SPRT.SC DAM.WATER-LIN/1-14N-NARROW .....0-7  
SPRT.SC DAM.WATER-LIN/1-4IN-TURBHI .....0-8  
SPRT.SC DAM.WATER-LIN/1-14N-TURBHI .....0-9  
SPRT.SC  
SPRT.SC DAM.WATER-LIN/2 .....0-11  
SPRT.SC DAM.WATER-LIN/2A .....0-12  
SPRT.SC DAM.WATER-LIN/2A-124F-PUR2 .....0-13  
SPRT.SC DAM.WATER-LIN/2A-124N-PUR2 .....0-14  
SPRT.SC DAM.WATER-LIN/2A-124N-PUR2 .....0-15  
SPRT.SC DAM.WATER-LIN/2A-14F-PUR2 .....0-16  
SPRT.SC DAM.WATER-LIN/2A-14N-PUR2 .....0-17  
SPRT.SC DAM.WATER-LIN/2A-14N-PUR2 .....0-18  
SPRT.SC DAM.WATER-LIN/2A-412F-PUR3 .....0-19  
SPRT.SC DAM.WATER-LIN/2A-14F-PUR3 .....0-20  
SPRT.SC DAM.WATER-LIN/2A-412F-PUR4 .....0-21  
SPRT.SC DAM.WATER-LIN/2A-14F-PUR4 .....0-22  
SPRT.SC  
SPRT.SC DAM.CHAN-CALIB/2B .....0-24  
SPRT.SC DAM.WATER-LIN/2B .....0-25  
SPRT.SC DAM.WATER-LIN/2B-14N-PUR1 .....0-26  
SPRT.SC DAM.WATER-LIN/2B-124N-MIX1 .....0-27  
SPRT.SC DAM.WATER-LIN/2B-124F-PUR2 .....0-28  
SPRT.SC DAM.WATER-LIN/2B-124N-PUR2 .....0-29  
SPRT.SC DAM.WATER-LIN/2B-124N-PUR2 .....0-30  
SPRT.SC DAM.WATER-LIN/2B-14F-PUR2 .....0-31  
SPRT.SC DAM.WATER-LIN/2B-14N-PUR2 .....0-32  
SPRT.SC DAM.WATER-LIN/2B-14N-PUR2 .....0-33  
SPRT.SC DAM.WATER-LIN/2B-412F-PUR3 .....0-34  
SPRT.SC DAM.WATER-LIN/2B-14F-PUR3 .....0-35  
SPRT.SC DAM.WATER-LIN/2B-412F-PUR4 .....0-36  
SPRT.SC DAM.WATER-LIN/2B-14F-PUR4 .....0-37  
SPRT.SC  
SPRT.SC DAM.CHAN-CALIB/2D .....0-39  
SPRT.SC DAM.WATER-LIN/2D .....0-40  
SPRT.SC DAM.WATER-LIN/2D-412F-PUR4 .....0-41  
SPRT.SC  
SPRT.SC DAM.WATER-LIN/3 .....0-43  
SPRT.SC DAM.WATER-LIN/3A .....0-44  
SPRT.SC DAM.WATER-LIN/3A-LIN-N1 .....0-45  
SPRT.SC  
SPRT.SC DAM.CHAN-CALIB/3C .....0-47  
SPRT.SC DAM.WATER-LIN/3C .....0-48  
SPRT.SC DAM.WATER-LIN/3C-LIN-N1 .....0-49  
SPRT.SC  
SPRT.SC DAM.WATER-NESS/1 .....0-51  
SPRT.SC DAM.WATER-NESS/2A .....0-52  
SPRT.SC DAM.WATER-NESS/2B .....0-53  
SPRT.SC DAM.WATER-NESS/2D .....0-54  
SPRT.SC DAM.WATER-NESS/3A .....0-55  
SPRT.SC DAM.WATER-NESS/3C .....0-56

DAM.EXP----

DAM.EXP----

DAM.EXP----

DAM.EXP----

DAM.EXP----

DAM.EXP----

**DAM PACKAGE APPENDIX 0  
SPECTRAL LIMITS/TRANSFORMS**

**APPENDIX-0  
002**

**SPRT.SC  
SPRT.SC DAM.TURBID-NESS/1 .....0-58  
SPRT.SC DAM.TURBID-NESS/2A .....0-59  
SPRT.SC DAM.TURBID-NESS/2B .....0-60**

**DAM.EXP----**

**DAM PACKAGE APPENDIX H  
SAMPLE RUNSTREAMS**

**APPENDIX-H  
001**

SPRT.SC DAM.	APPENDIX-H . (0009).....H-1
SPRT.SC DAM.EXP----	.
SPRT.SC DAM.EXP----	.
SPRT.SC DAM.RUN-ERTS-DUP	. ERTS-DUP.....H-4
SPRT.SC DAM.RUN-ERTS-DUP/JSC	. ERTS-DUP(JSC).....H-5
SPRT.SC DAM.EXP----	. .....
SPRT.SC DAM.RUN-PICTAB/X-SF	. PICTAB(SINGLE FILE 'X' TAPES)..H-7
SPRT.SC DAM.RUN-PICTAB/X-MF	. PICTAB(MULTI-FILE 'X' TAPES)...H-8
SPRT.SC DAM.RUN-PICTAB/PH	. PICTAB('PH' TAPES).....H-9
SPRT.SC DAM.EXP----	. .....
SPRT.SC DAM.RUN-CONTROL	. CONTROL.....H-11
SPRT.SC DAM.EXP----	. .....
SPRT.SC DAM.RUN-CLA-PRC/X	. CLASSIFY/PRTCLASS('X' TAPES)...H-13
SPRT.SC DAM.RUN-CLA-PRC/PH	. CLASSIFY/PRTCLASS('PH' TAPES)..H-14
SPRT.SC DAM.EXP----	. .....
SPRT.SC DAM.RUN-STATUS	. STATUS.....H-16

**DAM PACKAGE APPENDIX I  
REVISIONS AND NEWS**

**APPENDIX-I  
001**

**SPRT.SC DAM.APPENDIX-I  
SPRT.SC DAM.REV-DAM  
SPRT.SC DAM.NEW-DAM  
SPRT.SC DAM.NEW-PLTCLASS**

**DAN PACKAGE APPENDIX J  
DEFAULT COMMANDS**

**APPENDIX-J  
001**

**SPRT.SC DAN.APPENDIX-J  
SPRT.SC DAN.DEF-ERTSIDC  
SPRT.SC DAN.DEF-PICTAB  
SPRT.SC DAN.DEF-CONTROL  
SPRT.SC DAN.DEF-CLASSIFY  
SPRT.SC DAN.DEF-PRTDET  
SPRT.SC DAN.PRO-DEF-RAD  
SPRT.SC DAN.PRO-DEF-DEN  
SPRT.SC DAN.PRO-DEF-CLA  
SPRT.SC DAN.DEF-PRTCLASS  
SPRT.SC DAN.PRC-DEF-RAD  
SPRT.SC DAN.PRC-DEF-DEN  
SPRT.SC DAN.PRC-DEF-CLA  
SPRT.SC DAN.DEF-PLTCLASS  
SPRT.SC DAN.DEF-STATUS**

**DAM PACKAGE APPENDIX K  
SYSTEM IMPLEMENTATION**

**APPENDIX-K  
001**

**BPRT.SC DAM.PREFACE-K  
BPRT.SC DAM.APPENDIX-K  
BPRT.SC DAM.SYS-HIA  
BMSG.N DAM.SYS-TITLE  
BMSG.N DAM.SYS-LIST  
BPRT.SC DAM.SYS-EXPLAIN  
BPRT.SC DAM.SYS-COPYCOM  
BPRT.SC DAM.SYS-FOROPT  
BPRT.SC DAM.SYS-BLOCK  
BPRT.SC DAM.SYS-MAPOPT  
BPRT.SC DAM.SYS-COMPILE  
BPRT.SC DAM.SYS-COLLECT  
BPRT.SC DAM.SYS-DELETE  
BPRT.SC DAM.SYS-GENCOM**

- . HIERARCHY**
- . TITLE PAGE**
- . PRINT ANNOTATED LIST OF SYMBOLIC ELEMENTS**
- . IMPLEMENTATION INSTRUCTIONS**
- . COPY COMPILATION/COLLECTION COMMAND STREAMS TO TPFs**
- . STANDARD FORTRAN COMPILER OPTIONS**
- . BLOCK DATA SUBROUTINE**
- . STANDARD MAP PROCESSOR OPTIONS**
- . COMPILATION COMMAND STREAM (MUST NOT 3ADD FROM DAM)**
- . COLLECTION COMMAND STREAM (MUST NOT 3ADD FROM DAM)**
- . DELETE SOURCE SYMBOLICS (MUST NOT 3ADD FROM DAM)**
- . FOR JSC TO GENERATE DAM.SYS-COMPILE, DAM.SYS-DELETE**

DAM PACKAGE APPENDIX L  
MAIN PROGRAMS/ROUTINES

APPENDIX-L  
001

SPRT.SC	DAM.PREFACE-L	. (0009)	SET TABS 8 30 & 34
SPRT.S	DAM.APPENDIX-L		
MSO.N			
SPRT.S	DAM.SETUP	. SET UP TPFs AND TTY. AND PRINT NEWS	
SPRT.SC	DAM.SETUP-HIA	. HIERARCHY	
MSO.N			
SPRT.S	DAM.SETUP/N	. SET UP TPFs AND TTY. AND DON'T PRINT NEWS	
MSO.N			
SPRT.S	DAM.ERSPRTCN	. SUBMIT PRINT CONTROL SPECIFICATIONS	
SPRT.SC	DAM.ERSPRTCN-MAP	. COLLECTOR SYMBOLICS FOR REAL ABSOLUTE	
SPRT.SC	DAM.ERSPRTCN-MAP/VIRTUAL	. COLLECTOR SYMBOLICS FOR REAL ABS (NO VIRTUAL)	
MSO.N			
SPRT.S	DAM.DATA/CHECKOUT	. PSEUDO EXEC COMMAND: INITIATE DATA/CHECKOUT MODE	
SPRT.SC	DAM.DATA-MAP	. COLLECTOR SYMBOLICS FOR REAL ABSOLUTE	
SPRT.SC	DAM.DATA-MAP/VIRTUAL	. COLLECTOR SYMBOLICS FOR REAL ABS (NO VIRTUAL)	
SPRT.SC	DAM.DATA-CHECK	. DATA/CHECKOUT	
MSO.N			
SPRT.S	DAM.IDFILE	. PSEUDO EXEC COMMAND: IDENTIFY TAPE OR DISK FILE	
SPRT.SC	DAM.IDFILE-MAP	. COLLECTOR SYMBOLICS FOR REAL ABSOLUTE	
SPRT.SC	DAM.IDFILE-MAP/VIRTUAL	. COLLECTOR SYMBOLICS FOR REAL ABS (NO VIRTUAL)	
MSO.N			
SPRT.S	DAM.LOCATE	. PSEUDO EXEC COMMAND: POSITION MULTI-FILE TAPE	
SPRT.SC	DAM.LOCATE-MAP	. COLLECTOR SYMBOLICS FOR REAL ABSOLUTE	
SPRT.SC	DAM.LOCATE-MAP/VIRTUAL	. COLLECTOR SYMBOLICS FOR REAL ABS (NO VIRTUAL)	
MSO.N			
SPRT.S	DAM.USWAP	. PSEUDO EXEC COMMAND: SWAP TAPE DRIVE UNITS	
SPRT.SC	DAM.USWAP-MAP	. COLLECTOR SYMBOLICS FOR REAL ABSOLUTE	
SPRT.SC	DAM.USWAP-MAP/VIRTUAL	. COLLECTOR SYMBOLICS FOR REAL ABS (NO VIRTUAL)	
MSO.N			
SPRT.S	DAM.ERTS-DUP	. DUPLICATE ERTS MSS TAPE	
MSO.N			
SPRT.S	DAM.ERTSIDC	. PRINT ID/ANNOTATION DATA FROM ERTS MSS TAPES	
SPRT.SC	DAM.ERTSIDC-HIA	. HIERARCHY	
SPRT.SC	DAM.ERTSIDC/VIRTUAL	. SWAP DAM.ERTSIDC TO TPFs AND 3XQT.1	
SPRT.SC	DAM.ERTSIDC-MAP	. COLLECTOR SYMBOLICS FOR REAL ABSOLUTE	
SPRT.SC	DAM.ERTSIDC-MAP/VIRTUAL	. COLLECTOR SYMBOLICS FOR VIRTUAL ABSOLUTE	
MSO.N			
SPRT.S	DAM.PICTAB	. DISPLAY/TABULATE/FACTOR/PARTITION ERTS MSS DATA	
SPRT.SC	DAM.PICTAB-HIA	. HIERARCHY	
SPRT.SC	DAM.PICTAB/VIRTUAL	. SWAP DAM.PICTAB TO TPFs AND 3XQT.1	
SPRT.SC	DAM.PICTAB-MAP	. COLLECTOR SYMBOLICS FOR REAL ABSOLUTE	
SPRT.SC	DAM.PICTAB-MAP/VIRTUAL	. COLLECTOR SYMBOLICS FOR VIRTUAL ABSOLUTE	
SPRT.SC	DAM.PIC000	. CALL PHASE 0 (COMMAND) ROUTINES FOR PICTAB	
SPRT.SC	DAM.PIC129	. CALL PHASE 1/2/9 ROUTINES FOR PICTAB	
SPRT.SC	DAM.PIC345	. CALL PHASE 3/4/5 ROUTINES FOR PICTAB	
SPRT.SC	DAM.PIC678	. CALL PHASE 6/7/8 ROUTINES FOR PICTAB	
SPRT.SC	DAM.PIC019	. DISPLAY MSS-DERIVED DATA (PHASE 0)	
SPRT.SC	DAM.PICD13	. DISPLAY RADIANCE (PHASE 3)	
SPRT.SC	DAM.PICD14	. DISPLAY GRADIENT/LAPLACIAN/VARIANCE (PHASE 4)	
SPRT.SC	DAM.PIC015	. DISPLAY CLASS (PHASE 5)	
SPRT.SC	DAM.PICD19	. DISPLAY MSS-DERIVED DATA (PHASE 9)	
SPRT.SC	DAM.DISH19	. HISTOGRAM DISPLAYED DATA (PHASE 0)	
SPRT.SC	DAM.PICEX1	. TERMINATION ROUTINE (PHASE 0)	
SPRT.SC	DAM.PICFAC	. FACTOR MSS CHANNELS (PHASE 0)	
SPRT.SC	DAM.PICFA3	. FACTOR MSS CHANNELS (PHASE 3)	
SPRT.SC	DAM.PICFA9	. FACTOR MSS CHANNELS (PHASE 9)	



DAH PACKAGE APPENDIX L  
MAIN PROGRAMS/ROUTINES

APPENDIX-L  
002

SPRT.SC DAH.PICLI8	. LIST MSS-DERIVED DATA (PHASE 0)
SPRT.SC DAH.PICLI3	. LIST RADIANCE (PHASE 3)
SPRT.SC DAH.PICLI4	. LIST GRADIENT (PHASE 4)
SPRT.SC DAH.PICLI5	. LIST CLASS (PHASE 5)
SPRT.SC DAH.PICLI9	. LIST MSS-DERIVED DATA (PHASE 9)
SPRT.SC DAH.PICPAR	. PARTITION FACTOR SPACE (PHASE 0)
SPRT.SC DAH.PICPA3	. PARTITION BY DENSITY (PHASE 3)
SPRT.SC DAH.PICPA4	. PARTITION BY GRADIENT/LAPLACIAN/VARIANCE (PM4)
SPRT.SC DAH.PICPA8	. PARTITION FACTOR SPACE (PHASE 8)
SPRT.SC DAH.PICPA9	. PARTITION FACTOR SPACE (PHASE 9)
SPRT.SC DAH.PICPIC	. PICTURE MSS-DERIVED DATA (PHASE 0)
SPRT.SC DAH.PICPI3	. PICTURE RADIANCE (PHASE 3)
SPRT.SC DAH.PICPI4	. PICTURE GRADIENT/LAPLACIAN/VARIANCE (PHASE 4)
SPRT.SC DAH.PICPI5	. PICTURE CLASS (PHASE 5)
SPRT.SC DAH.PICPI9	. PICTURE MSS-DERIVED DATA (PHASE 9)
SPRT.SC DAH.PICPRO	. PROFILE MSS-DERIVED DATA (PHASE 0)
SPRT.SC DAH.PICPR3	. PROFILE MSS-DERIVED DATA (PHASE 3)
SPRT.SC DAH.PICPR9	. PROFILE MSS-DERIVED DATA (PHASE 9)
SPRT.SC DAH.PICROT	. ROTATE FACTOR STRUCTURE/COEFFICIENTS
SPRT.SC DAH.PICTOT	. TOTAL TABULATIONS
SPRT.SC DAH.PICTO3	. TOTAL TABULATIONS
SPRT.SC DAH.PICTO9	. TOTAL TABULATIONS
SPRT.SC DAH.PICXQT	. INITIALIZATION ROUTINE (PHASE 0)
SPRT.SC DAH.IDCPIC	. IDENTIFY CURRENT COMMAND SPECS (UTILITY)
SPRT.SC DAH.OPRPIC	. OPEN ALTERNATE PRINT FILES (UTILITY)
SHSG.N	
SPRT.S DAH.CONTROL	. ADJUST/DIAGRAM CONTROL NETWORK
SPRT.SC DAH.CONTROL-MIA	. HIERARCHY
SPRT.SC DAH.CONTROL/VIRTUAL	. SHAP DAH.CONTROL TO TPFs AND SXQT.1
SPRT.SC DAH.CONTROL-MAP	. COLLECTOR SYMBOLICS FOR REAL ABSOLUTE
SPRT.SC DAH.CONTROL-MAP/VIRTUAL	. COLLECTOR SYMBOLICS FOR VIRTUAL ABSOLUTE
SPRT.SC DAH.CON000	. CALL PHASE 0 (COMMAND) ROUTINES FOR CONTROL
SPRT.SC DAH.CONADJ	. ADJUST NETWORK (PHASE 0)
SPRT.SC DAH.CONDIA	. DIAGRAM NETWORK (PHASE 0)
SPRT.SC DAH.CONEXI	. TERMINATION ROUTINE (PHASE 0)
SPRT.SC DAH.CONXQT	. INITIALIZATION ROUTINE (PHASE 0)
SPRT.SC DAH.DIAERR	. DIAGRAM ERRORS
SPRT.SC DAH.DLSTSQ	. LEAST SQUARES 81-LINEAR FIT
SHSG.N	
SPRT.S DAH.CLASSIFY	. CLASSIFY DATA ON ERTS MSS TAPE
SPRT.SC DAH.CLASSIFY-MIA	. HIERARCHY
SPRT.SC DAH.CLASSIFY/VIRTUAL	. SHAP DAH.CLASSIFY TO TPFs AND SXQT.1
SPRT.SC DAH.CLASSIFY-MAP	. COLLECTOR SYMBOLICS FOR REAL ABSOLUTE
SPRT.SC DAH.CLASSIFY-MAP/VIRTUAL	. COLLECTOR SYMBOLICS FOR VIRTUAL ABSOLUTE
SPRT.SC DAH.CLA000	. CALL PHASE 0 (COMMAND) ROUTINES FOR CLASSIFY
SPRT.SC DAH.CLA129	. CALL PHASE 1/2/9 ROUTINES FOR CLASSIFY
SPRT.SC DAH.CLA345	. CALL PHASE 3/4/5 ROUTINES FOR CLASSIFY
SPRT.SC DAH.CLADET	. GENERATE DETECTION FILE (PHASE 0)
SPRT.SC DAH.CLADE3	. GENERATE RADIANCE DETECTION FILE (PHASE 3)
SPRT.SC DAH.CLADE4	. GENERATE DENSITY DETECTION FILE (PHASE 4)
SPRT.SC DAH.CLADE5	. GENERATE CLASS DETECTION FILE (PHASE 5)
SPRT.SC DAH.CLADE9	. GENERATE DETECTION FILE (PHASE 9)
SPRT.SC DAH.CLAEXI	. TERMINATION ROUTINE (PHASE 0)
SPRT.SC DAH.CLARAD	. GET/CHECK RADIANCE LIMITS (PHASE 0)
SPRT.SC DAH.CLAXQT	. INITIALIZATION ROUTINE (PHASE 0)
SPRT.SC DAH.CLSO2N	. CLOSE OUTPUT DETECTION FILE (UTILITY)

```

SPRT.SC DAH.OPNOEN      . OPEN OUTPUT DETECTION FILE (UTILITY)
SPRT.SC DAH.OPRCLA      . OPEN ALTERNATE PRINT FILE (UTILITY)
SPRT.SC DAH.SLMCLA      . PLOT SPECTRAL LIMITS (UTILITY)
SNSO.N
SPRT.S  DAH.PRTOET      . DISPLAY DATA FROM DETECTION FILE(S)
SPRT.SC DAH.PRTOET-HIA  . HIERARCHY
SPRT.SC DAH.PRTOET/VIRTUAL . SHAP DAH.PRTOET TO TPFS AND SXQT.1
SPRT.SC DAH.PRTOET-MAP   . COLLECTOR SYMBOLICS FOR REAL ABSOLUTE
SPRT.SC DAH.PRTOET-MAP/VIRTUAL . COLLECTOR SYMBOLICS FOR VIRTUAL ABSOLUTE
SPRT.SC DAH.PRO000      . CALL PHASE 0 (COMMAND) ROUTINES FOR PRTOET
SPRT.SC DAH.PRO129      . CALL PHASE 1/2/9 ROUTINES FOR PRTOET
SPRT.SC DAH.PRO345      . CALL PHASE 3/4/5 ROUTINES FOR PRTOET
SPRT.SC DAH.PRO015      . DISPLAY DETECTION DATA (PHASE 0)
SPRT.SC DAH.PRO013      . DISPLAY DETECTION DATA (PHASE 3)
SPRT.SC DAH.PRO019      . DISPLAY DETECTION DATA (PHASE 9)
SPRT.SC DAH.PROEX1      . TERMINATION ROUTINE
SPRT.SC DAH.PROL15      . LIST DETECTION DATA (PHASE 0)
SPRT.SC DAH.PROL13      . LIST DETECTION DATA (PHASE 3)
SPRT.SC DAH.PROL19      . LIST DETECTION DATA (PHASE 9)
SPRT.SC DAH.PROPIC      . PICTURE DETECTION DATA (PHASE 0)
SPRT.SC DAH.PROPI3      . PICTURE DETECTION DATA (PHASE 3)
SPRT.SC DAH.PROPI9      . PICTURE DETECTION DATA (PHASE 9)
SPRT.SC DAH.PROXQT      . INITIALIZATION ROUTINE
SPRT.SC DAH.IDCPRD      . IDENTIFY CURRENT COMMAND SPECS (UTILITY)
SPRT.SC DAH.OPRPRD      . OPEN ALTERNATE PRINT FILES (UTILITY)
SNSO.N
SPRT.S  DAH.PRTCLASS    . OUTPUT CLASSIFIED ERTS MAPS ON LINE PRINTER
SPRT.SC DAH.PRTCLASS-HIA . HIERARCHY
SPRT.SC DAH.PRTCLASS/VIRTUAL . SHAP DAH.PRTCLASS TO TPFS AND SXQT.1
SPRT.SC DAH.PRTCLASS-MAP   . COLLECTOR SYMBOLICS FOR REAL ABSOLUTE
SPRT.SC DAH.PRTCLASS-MAP/VIRTUAL . COLLECTOR SYMBOLICS FOR VIRTUAL ABSOLUTE
SPRT.SC DAH.PRC000      . CALL PHASE 0 (COMMAND) ROUTINES FOR PRTCLASS
SPRT.SC DAH.PRCX1      . TERMINATION ROUTINE (PHASE 0)
SPRT.SC DAH.PRCMAP      . MAP RADIANCE/DENSITY/CLASS (PHASE 0)
SPRT.SC DAH.PRCXQT      . INITIALIZATION ROUTINE (PHASE 0)
SPRT.SC DAH.HAPRNT      . PRINT MAPS
SPRT.SC DAH.NITH00      . PROVIDE UNIT HEADING (UTILITY)
SPRT.SC DAH.OPRPRC      . OPEN ALTERNATE PRINT FILES (UTILITY)
SPRT.SC DAH.RESPRC      . RESAMPLE DETECTION PIXELS (UTILITY)
SNSO.N
SPRT.S  DAH.PLTCLASS    . OUTPUT CLASSIFIED ERTS MAPS ON PEN PLOTTER
SPRT.SC DAH.PLTCLASS/VIRTUAL . SHAP DAH.PLTCLASS TO TPFS AND SXQT.1
SPRT.SC DAH.PLTCLASS-MAP   . COLLECTOR SYMBOLICS FOR REAL ABSOLUTE
SPRT.SC DAH.PLTCLASS-MAP/VIRTUAL . COLLECTOR SYMBOLICS FOR VIRTUAL ABSOLUTE
SPRT.SC DAH.PLC000      . CALL PHASE 0 (COMMAND) ROUTINES FOR PLTCLASS
SPRT.SC DAH.PLC129      . CALL PHASE 1/2/9 ROUTINES FOR PLTCLASS
SPRT.SC DAH.PLCX1      . TERMINATION ROUTINE (PHASE 0)
SPRT.SC DAH.PLCMAP      . MAP RADIANCE/DENSITY/CLASS (PHASE 0)
SPRT.SC DAH.PLCXQT      . INITIALIZATION ROUTINE (PHASE 0)
SPRT.SC DAH.ITICPL      . GENERATE & PLOT INTERIOR TICKS (UTILITY)
SPRT.SC DAH.HTICPL      . GENERATE & PLOT MARGINAL TICKS (UTILITY)
SPRT.SC DAH.NITHPL      . PLOT UNIT HEADING (UTILITY)
SPRT.SC DAH.OPLPLC      . OPEN ALTERNATE PLOT FILES (UTILITY)
SPRT.SC DAH.REOPLC      . REGISTER DETECTION PIXELS (UTILITY)
SNSO.N
SPRT.S  DAH.FLMCLASS    . OUTPUT CLASSIFIED ERTS MAPS ON FILM RECORDER

```

DAM PACKAGE APPENDIX L  
MAIN PROGRAMS/ROUTINES

APPENDIX-L  
004

SMSG.N	
SPRT.S	DAM.STATUS . DETERMINE STATUS OF DAM PACKAGE RUNS
SPRT.SC	DAM.STATUS-HIA . HIERARCHY
SPRT.SC	DAM.STATUS-VIRTUAL . SHAP DAM.STATUS TO TPFS & SXQT.I
SPRT.SC	DAM.STATUS-MAP . COLLECTOR SYMBOLICS FOR REAL ABSOLUTE
SPRT.SC	DAM.STATUS-MAP/VIRTUAL . COLLECTOR SYMBOLICS FOR VIRTUAL ABSOLUTE
SMSG.N	
SPRT.S	DAM.DITCOP . DISK TO TAPE COPY PROGRAM
SPRT.SC	DAM.DITCOP/VIRTUAL . SHAP DAM.DITCOP TO TPFS AND SXQT.I
SPRT.SC	DAM.DITCOP-MAP . COLLECTOR SYMBOLICS FOR REAL ABSOLUTE
SPRT.SC	DAM.DITCOP-MAP/VIRTUAL . COLLECTOR SYMBOLICS FOR VIRTUAL ABSOLUTE
SPRT.SC	DAM.DIT000 . CALL PHASE 0 (COMMAND) ROUTINES FOR DITCOP
SPRT.SC	DAM.DITEXI . TERMINATION ROUTINE
SPRT.SC	DAM.DITDUP . DUPLICATE DETECTION FILE ONTO TAPE FROM DISK
SPRT.SC	DAM.DITVER . VERIFY DETECTION FILE ON TAPE
SPRT.SC	DAM.DITXQT . INITIALIZATION ROUTINE

**DAM PACKAGE APPENDIX M  
COMMAND ROUTINES**

**APPENDIX-M  
001**

SPRT.SC DAM.PREFACE-M	. (0005)	SET TABS 8 13 & 26
SPRT.SC DAM.APPENDIX-M	.	
SHSO.N ...ADJ	. ADJUST NETWORK (SEE APPENDIX L)	
SPRT.SC DAM.KMDALI	. ALIGN COORDINATE SYSTEMS	
SPRT.SC DAM.KMDATT	. GET/CHECK PLATFORM ATTITUDE	
SPRT.SC DAM.KMDCCN	. GET/CHECK SCENE CENTER SCAN COORDINATES	
SPRT.SC DAM.KMOCNA	. GET/CHECK RAW/TRANSFORMED CHANNEL(S)	
SPRT.SC DAM.DETCHA	. GET/CHECK DETECTION CHANNEL(S)	
SPRT.SC DAM.KMDCLE	. CLEAR WARNINGS/ERRORS	
SPRT.SC DAM.KMDCOL	. GET/CHECK COLOR(S)	
SPRT.SC DAM.KMDCOP	. GET/CHECK NUMBER OF OUTPUT COPIES	
SPRT.SC DAM.KMDCOU	. GET/CHECK COUNT PER PIXEL	
SPRT.SC DAM.KMDCRO	. CROSSTABULATE	
SPRT.SC DAM.KMDDEN	. GET/CHECK DENSITY LIMITS	
SHSO.N ...DET	. DETECT (SEE APPENDIX L)	
SHSO.N ...DIA	. DIAGRAM NETWORK (SEE APPENDIX L)	
SHSO.N ...DIS	. DISPLAY ON ALPHA-NUMERIC DEVICE (SEE APPENDIX L)	
SHSO.N ...EXI	. EXIT (SEE APPENDIX L)	
SPRT.SC DAM.KMDEXP	. EXPLAIN PROGRAM/COMMAND	
SHSO.N ...FAC	. FACTOR (SEE APPENDIX L)	
SPRT.SC DAM.KMDFI	. END IF ... FI BLOCK	
SPRT.SC DAM.KMDGEO	. GET/CHECK SCENE GEOMETRY	
SPRT.SC DAM.KMDHEA	. GET/CHECK PAGE HEADINGS	
SHSO.N ...HIS	. HISTOGRAM (SEE APPENDIX L)	
SPRT.SC DAM.KMDIF	. BEGIN IF ... FI BLOCK	
SPRT.SC DAM.KMDINT	. GET/CHECK INTENSITY(S)	
SPRT.SC DAM.KMDLIN	. GET/CHECK LINEAR HEIGHTS/GAIN/BIAS	
SHSO.N ...LIS	. LIST (SEE APPENDIX L)	
SHSO.N ...MAP	. MAP (SEE APPENDIX L)	
SPRT.SC DAM.KMDHER	. GET/CHECK TRANSVERSE MERCATOR CENTRAL MERIDIAN	
SHSO.N ...MOD	. MODEL (SEE APPENDIX L)	
SPRT.SC DAM.KMDNAM	. GET/CHECK MATERIAL NAME	
SPRT.SC DAM.KMDNEH	. PRINT NEWS OF PROGRAM CHANGES	
SPRT.SC DAM.KMDNEX	. SPECIFY CONDITION FOR PERFORMING NEXT COMMAND	
SPRT.SC DAM.KMDOFF	. TURN OFF MODE SWITCH(ES)	
SPRT.SC DAM.KMDON	. TURN ON MODE SWITCH(ES)	
SPRT.SC DAM.KMDORI	. GET/CHECK WINDOW ORIGIN	
SPRT.SC DAM.KMDPAG	. SKIP TO TOP OF NEXT PAGE	
SHSO.N ...PAR	. PARTITION (SEE APPENDIX L)	
SPRT.SC DAM.KMDPEE	. 'PEEK' INTO LABELLED COMMONS (FOR DEBUGGING)	
SHSO.N ...PIC	. PICTURE ON COLOR CRT (SEE APPENDIX L)	
SPRT.SC DAM.KMDPLO	. GET/CHECK PLOTTER SPECIFICATIONS	
SPRT.SC DAM.KMDPOI	. GET/CHECK CONTROL/CHECK POINT	
SPRT.SC DAM.KMDPOK	. 'POKE' (CHANGE) LABELLED COMMONS (FOR DEBUGGING)	
SPRT.SC DAM.KMDPOL	. GET/CHECK POLAR GAIN/BIAS	
SPRT.SC DAM.KMDPRI	. GET/CHECK PRINTER SPECIFICATIONS	
SHSO.N ...PRO	. PROFILE (SEE APPENDIX L)	
SPRT.SC DAM.KMDRAD	. GET/CHECK RADIANCE LIMITS	
SHSO.N ...RAD	. (SEE ALSO APPENDIX L)	
SPRT.SC DAM.KMDRAN	. GET/CHECK RANKING(S)	
SPRT.SC DAM.KMDREN	. GET/CHECK NEW WINDOW SEQUENCE NUMBER	
SPRT.SC DAM.KMDRES	. GET/CHECK RESAMPLING SPECIFICATIONS	
SHSO.N ...ROT	. ROTATE (SEE APPENDIX L)	
SPRT.SC DAM.KMDSCA	. GET/CHECK WINDOW SCALE	
SPRT.SC DAM.KMDSCE	. GET/CHECK SCENE NUMBER AND SIZE	
SPRT.SC DAM.KMDSCR	. GET/CHECK SCRIPT	

**DAM PACKAGE APPENDIX M  
COMMAND ROUTINES**

**APPENDIX-M  
002**

SPRT.SC DAM.KNDSHA	. GET/CHECK SHARPENING FILTER COEFFICIENTS
SPRT.SC DAM.KNDISZ	. GET/CHECK SCENE SIZE IN SCAN COORDINATES
SPRT.SC DAM.KNDSPA	. GET/CHECK WINDOW SPACING
SPRT.SC DAM.KNDSPH	. GET/CHECK SPHEROID GEODETIC PARAMETERS
SPRT.SC DAM.KNDSYN	. GET/CHECK SYMBOL(S)
SPRT.SC DAM.KNDTAB	. TABULATE
SPRT.SC DAM.KNDTIC	. GET/CHECK TICK INTERVALS
SPRT.SC DAM.KNDTIM	. PRINT CLOCK TIME & CHARGE TIME
SPRT.SC DAM.KNDTIN	. GET/CHECK TINT SPECIFICATIONS
SPRT.SC DAM.KNDTOL	. GET/CHECK TOLERANCE
SPRT.SC DAM.KNDWIN	. GET/CHECK WINDOW VERTICES
SPRT.SC DAM.KNDXXX	. GET/CHECK/PROCESS MACRO COMMAND (CALLS KMXXED)
SPRT.SC DAM.KNDZON	. GET/CHECK UTM PROJECTION ZONE
SPRT.SC DAM.KNDQAD	. SADD -- DYNAMIC SADD
SPRT.SC DAM.KNDQAS	. SASO -- DYNAMIC SASO
SPRT.SC DAM.KNDQBR	. SBKPT -- DYNAMIC SBKPT
SPRT.SC DAM.KNDQFR	. SFREE -- DYNAMIC SFREE
SPRT.SC DAM.KNDQLO	. SLOG -- DYNAMIC SLOG
SPRT.SC DAM.KMXXED	. EDIT ACTUAL SPECS INTO MACRO COMMAND DEFINITION
SPRT.SC DAM.KMXXOS	. GET/EVALUATE ACTUAL SPEC FOR MACRO COMMAND

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**PREFACE-N  
001**

**PREFACE TO APPENDIX N  
-----**

**THIS APPENDIX CONTAINS DAN PACKAGE GENERAL UTILITY SUBROUTINES AND FUNCTIONS .**

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**APPENDIX-N  
001**

SPRT.SC DAN.PREFACE-N  
SPRT.SC DAN.APPENDIX-N  
SPRT.SC DAN.ARG77  
SPRT.SC DAN.ARGRET  
SPRT.SC DAN.ATRACE/DAN  
SPRT.SC DAN.BOX-CHR  
SPRT.SC DAN.BY'DMP  
SPRT.SC DAN.CALCHA  
SPRT.SC DAN.CALCOL  
SPRT.SC DAN.CALSCA  
SPRT.SC DAN.CALSPA  
SPRT.SC DAN.CALSYM  
SPRT.SC DAN.CALWIN  
SPRT.SC DAN.CLOSE3  
SPRT.SC DAN.CLOSE4  
SPRT.SC DAN.CLOSPR  
SPRT.SC DAN.CLSDMO  
SPRT.SC DAN.CL3BIP  
PRT.SC DAN.CORLT  
SPRT.SC DAN.CROPOW  
SPRT.SC DAN.DCORLT  
SPRT.SC DAN.DEO  
SPRT.SC DAN.DELETE-DENS  
SPRT.SC DAN.DDJR/MATHPACK  
SPRT.SC DAN.DOPCNT  
SPRT.SC DAN.DORECP  
SPRT.SC DAN.DOREQT  
SHSO.N .DLETPR  
SHSO.N .DLETCN  
SPRT.SC DAN.DMPTIC  
SPRT.SC DAN.DMPWIN  
SPRT.SC DAN.DSSPR  
SPRT.SC DAN.DSSPR3  
SPRT.SC DAN.DZDMS  
SPRT.SC DAN.EISRTD  
SPRT.SC DAN.ENVORI  
SPRT.SC DAN.ENVWIN  
SPRT.SC DAN.EOF  
SPRT.SC DAN.FACPRY  
SPRT.SC DAN.FLINFO  
SPRT.SC DAN.GCERY  
SPRT.SC DAN.GCHOM  
SHSO.N .GCLCC  
SPRT.SC DAN.GCONST  
SHSO.N .GCP5  
SHSO.N .GCSOM  
SHSO.N .GCUYM  
SPRT.SC DAN.GENTIC  
SPRT.SC DAN.GETDSR  
SPRT.SC DAN.GETRAD  
SPRT.SC DAN.GETS  
SHSO.N .GETSAL  
SHSO.N .GETSFR  
SHSO.N .GETSIN  
SHSO.N .GETSKM  
SHSO.N .GETSRL

. (0009) SET TABS 8 12 & 31  
.  
. DUMP A-REGISTER IN OCTAL (FOR DEBUGGING)  
. DETERMINE # OF ACTUAL ARGUMENTS & RETURN VECTOR  
. TRACE CALLS TO ASSEMBLER ROUTINES  
. BOX CHARACTERS  
. EXTRACT/SCALE/DUMP BYTE FIELDS (2'S COMPL)  
. CALIBRATE CHANNEL POINTERS  
. CALIBRATE COLOR/INTENSITY IN SYMBOL TABLE  
. CALIBRATE PRINT/PLOT COEFFICIENTS FOR SCALE  
. CALIBRATE PRINT/PLOT COEFFICIENTS FOR SPACING  
. CALIBRATE SYMBOL TABLE FOR PRINTING  
. CALIBRATE WINDOW ENVELOPES  
. CLOSE UNIT 3 (INPUT ERTS MSS DATA)  
. CLOSE COMMAND RECALL FILE (UNIT 4)  
. CLOSE AND PRINT ALTERNATE PRINT (SPOOL) FILES  
. WRITE COMMON ID/CLASSIFICATION HEADING  
. CLOSE UNIT 3 (INPUT ERTS MSS DATA IN BIP FORMAT)  
. CORRELATIONS/MEANS/DEVS FROM SUMS/SUMS-OF-PROD  
. CROP MSS OUTPUT WINDOW TO FIT PRINT FILE SIZE  
. CORRELATIONS/MEANS/DEVS FM D P SUMS/SUMS-OF-PROD  
. CONVERT DEGREES.MINUTES.SECONDS TO DEGREES  
. DELETE ALL DENSITY FILES  
. DBL PRECISION GAUSS-JORDAN REDUCTION  
. COMPUTE % OF TRACE FOR MATRIX DIAGONAL ELEMENTS  
. RECIPROCAL OF MATRIX DIAGONAL ELEMENTS  
. SQUARE ROOT OF MATRIX DIAGONAL ELEMENTS  
. DELETE ALTERNATE PRINT FILES (SEE CLOSPR)  
. DELETE INPUT DETECTION FILES (SEE OPNIZN)  
. DUMP TICK TABLE  
. DUMP WINDOW PACKET  
. COMPUTE DBL PRECISION SUMS & SUMS-OF-PRODUCTS  
. DBL PREC SUMS & SUMS-OF-PRODUCTS FROM UNIT 3  
. DEGREES TO DEGREES. MINUTES. SECONDS  
. SORT E-VALUES/E-VECTORS BY DESCENDING E-VALUES  
. ADD ORIGIN TO ENVELOPE (REAL WINDOW PACKET)  
. COMPUTE ENVELOPE FOR REAL WINDOW PACKET  
. END-OF-FILE (EOF)  
. PRINT FACTOR STRUCTURE/COEFFICIENTS/MEANS  
. GET FILE DESCRIPTIVE INFORMATION  
. LOAD 'ERT' GEOMETRIC CONSTANTS  
. LOAD 'MON' GEOMETRIC CONSTANTS  
. LOAD 'LCC' GEOMETRIC CONSTANTS (SEE GCHOM)  
. LOAD PROPER GEOMETRIC CONSTANTS  
. LOAD 'PS' GEOMETRIC CONSTANTS (SEE GCHOM)  
. LOAD 'SON' GEOMETRIC CONSTANTS (SEE GCHOM)  
. LOAD 'UTH' GEOMETRIC CONSTANTS (SEE GCHOM)  
. OPERATE/STORE/LIST/BOX TICKS  
. GET DISK SYMBOLIC RECORD  
. GET ORIGINAL/TRANSFORMED RADIANCE FROM ERTS TAPE  
. GET REMAINING CONTENTS OF UNIT 5 BUFFER  
. GET ALPHA DATA FIELD FROM UNIT 5 (SEE GETS)  
. GET FRACTION DATA FIELD FROM UNIT 5 (SEE GETS)  
. GET INTEGER DATA FIELD FROM UNIT 5 (SEE GETS)  
. GET CHARACTER DATA FIELD FROM UNIT 5 (SEE GETS)  
. GET REAL DATA FIELD FROM UNIT 5 (SEE GETS)

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

APPENDIX-M  
002

SHSO.N .GETSX  
 SPRT.SC DAN.MDUNIT  
 SPRT.SC DAN.ID4390  
 SPRT.SC DAN.IDERT  
 SPRT.SC DAN.IDERTS  
 SPRT.SC DAN.IDLUS  
 SPRT.SC DAN.IDUP  
 SHSO.N .INSTAT  
 SPRT.SC DAN.INVORI  
 SPRT.SC DAN.INVHIN  
 SPRT.SC DAN.IVKOLR  
 SPRT.SC DAN.JACHX/MATHPACK  
 SPRT.SC DAN.JOINZN  
 SPRT.SC DAN.KOLR41  
 SPRT.SC DAN.KSPRED  
 SPRT.SC DAN.LBOX41  
 SHSO.N .LDREON  
 SPRT.SC DAN.LDREOB  
 SPRT.SC DAN.LOCDSF  
 SPRT.SC DAN.LOOZ  
 SPRT.SC DAN.MAPHDO  
 SPRT.SC DAN.MATPRT  
 SHSO.N .MDCLRF  
 SHSO.N .MDCLRW  
 SHSO.N .MDFATL  
 SPRT.SC DAN.MDLOG  
 SHSO.N .MDNOTE  
 SHSO.N .MDWARN  
 SPRT.SC DAN.MSKPIX  
 SPRT.SC DAN.MVCONF  
 SPRT.SC DAN.MXHLT/MATHPACK  
 SPRT.SC DAN.NEOPIC  
 SPRT.SC DAN.NTABS/DAM  
 SPRT.SC DAN.NULSUB  
 SPRT.SC DAN.NVIATO  
 SPRT.SC DAN.OPENPR  
 SPRT.SC DAN.OPEN3  
 SHSO.N .OPEN4  
 SPRT.SC DAN.OPN12N  
 SPRT.SC DAN.OP3BIP  
 SPRT.SC DAN.OP3DSK  
 SPRT.SC DAN.OP3HDP  
 SPRT.SC DAN.OP3TAP  
 SPRT.SC DAN.OJANCL  
 SPRT.SC DAN.OJANOT  
 SPRT.SC DAN.O3HOR  
 SPRT.SC DAN.O3SZAM  
 SPRT.SC DAN.O3SZAR  
 SPRT.SC DAN.O3SZPH  
 SPRT.SC DAN.O3SZPR  
 SPRT.SC DAN.O3TOR  
 SHSO.N .PADORT  
 SPRT.SC DAN.PITROL  
 SPRT.SC DAN.PRONUM  
 SPRT.SC DAN.PROVFI  
 SPRT.SC DAN.PROVSY  
 . GET SEX'Y DATA FIELD FROM UNIT 5 (SEE GETS)  
 . PRINT HEADING LINE(S)  
 . ALLOCATE ARRAY OF 4390 WORDS IN 1-BANK  
 . PRINT SHORT ERTS SCENE IDENTIFICATION  
 . PRINT COMPLETE ERTS SCENE IDENTIFICATION  
 . PRINT SHORT ID FOR LOGICAL UNIT 3  
 . INTEGER DUPLICATE (INDIRECT REF TO OPTIONAL ARG)  
 . GET FORTRAN I/O STATUS (UNIVAC SYSTEM ROUTINE)  
 . ADD ORIGIN TO ENVELOPE (INTEGER WINDOW PACKET)  
 . COMPUTE ENVELOPE FOR INTEGER WINDOW PACKET  
 . INTEGER-COLOR-EQUIVALENT FOR COLOR  
 . JACOBI ITERATION TO FIND EIGEN- VALUES/VECTORS  
 . JOIN BUFFERS FROM TWO DETECTION FILES  
 . COLOR FOR INTEGER-COLOR-EQUIVALENT  
 . SPREAD COUNT FLAGS INTO INTERIOR UNDEFINED PIXELS  
 . LINE OF BOX DIGIT FOR INTEGER  
 . LOAD NOMINAL REGISTRATION PARAMETERS (SEE LDREOB)  
 . LOAD EXACT REGISTRATION PARAMETERS FROM UNIT 8  
 . LOCATE DISK SYMBOLIC FILE OR ELEMENT  
 . LOGARITHM, BASE 2 (TRUNCATED) OF INTEGER  
 . WRITE MAP WINDOW HEADING  
 . PRINT MATRIX  
 . CLEAR 'FATAL ERROR' COUNT (SEE MDLOG)  
 . CLEAR 'WARNING' COUNT (SEE MDLOG)  
 . PRINT/LOG/COUNT 'FATAL ERROR' (SEE MDLOG)  
 . LOG DIAGNOSTIC MESSAGES  
 . PRINT/LOG 'NOTE' (SEE MDLOG)  
 . PRINT/LOG/COUNT 'WARNING' (SEE MDLOG)  
 . MASK PIXELS IN BUFFER OUTSIDE NON-TRIVIAL WINDOW  
 . MOVE CONTENTS BETWEEN SPECIFIED LOCATIONS  
 . MATRIX MULTIPLICATION  
 . CONVERT DIGITAL PICTURE FROM POSITIVE TO NEGATIVE  
 . I/O UNIT NUMBER TABLE  
 . DO ABSOLUTELY NOTHING!!  
 . NAME 'VIA' 'TO' ROUTINES  
 . OPEN ALTERNATE PRINT (SPOOL) FILES (UNITS 10-19)  
 . OPEN UNIT 3 (INPUT ERTS MSS DATA)  
 . OPEN COMMAND RECALL FILE (SEE WRITE4)  
 . OPEN INPUT DETECTION FILE(S) (UNITS 21-24)  
 . OPEN UNIT 3 (INPUT ERTS MSS DATA IN BIP FORMAT)  
 . OPEN UNIT 3 (INPUT DATA ON DISK IN PKDEF FMT)  
 . OPEN UNIT 3 (INPUT ERTS MSS DATA IN HDP FORMAT)  
 . OPEN UNIT 3 (INPUT ERTS MSS DATA ON TAPE)  
 . OPEN UNIT 2 (HDP FORMAT ANCILLARY RECORDS)  
 . OPEN UNIT 3 (HDP FORMAT ANNOTATION RECORDS)  
 . OPEN UNIT 3 (HDP FORMAT HEADER RECORD)  
 . OPEN UNIT 3 (HDP AM FORMAT: SIZE 6 INPUT WINDOW)  
 . OPEN UNIT 3 (HDP AN FORMAT: SIZE 6 INPUT WINDOW)  
 . OPEN UNIT 3 (HDP PH FORMAT: SIZE 6 INPUT WINDOW)  
 . OPEN UNIT 3 (HDP PR FORMAT: SIZE 6 INPUT WINDOW)  
 . OPEN UNIT 3 (HDP FORMAT TAPE DIRECTORY RECORD)  
 . PROGRAM ABORT (SEE PSTOP)  
 . ESTIMATE PITCH AND ROLL  
 . PRINT BOX NUMBERS (VARIABLE HEIGHT)  
 . PRINT/OVERPRINT FILES  
 . PRINT/OVERPRINT SYMBOL BUFFER



**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**APPENDIX-N  
003**

SPRT.SC DAM.PRSYHL	. PRINT SYMBOL LEGEND
SPRT.SC DAM.PRTCHR	. WRITE BOX CHARACTERS ON ANY UNIT
SPRT.SC DAM.PRTINC	. SET PRINT LINES PER INCH
SPRT.SC DAM.PRTHRO	. SET PRINT MARGINS & LINES PER PAGE
SPRT.SC DAM.PSTART	. PROGRAM INITIATION
SPRT.SC DAM.PSTOP	. PROGRAM TERMINATION
SPRT.SC DAM.PXBONP	. DUMP PXBDEF PIXEL BUFFER PREAMBLE (FOR DEBUGGING)
SPRT.SC DAM.PX4AM	. PXBDEF PREAMBLE FOR MOP 'AM' PREAMBLE
SPRT.SC DAM.PX4AR	. PXBDEF PREAMBLE FOR MOP 'AR' PREAMBLE
SPRT.SC DAM.PX4PH	. PXBDEF PREAMBLE FOR MOP 'PH' PREAMBLE
SPRT.SC DAM.PX4PR	. PXBDEF PREAMBLE FOR MOP 'PR' PREAMBLE
SPRT.SC DAM.QUAD	. $FIT\ Y = A \cdot X^2 + B \cdot X + C$ & SOLVE FOR EXTREMUM
SPRT.SC DAM.QUARTN	. NORMALIZED QUARTIMAX ROTATION CRITERION
SPRT.SC DAM.QUARTU	. UN-NORMALIZED QUARTIMAX ROTATION CRITERION
SPRT.SC DAM.RD3BIL	. READ UNIT 3 (ERTS MSS DATA IN MOP BIL FORMAT)
SPRT.SC DAM.RD3BIP	. READ UNIT 3 (ERTS MSS DATA IN BIP FORMAT)
SPRT.SC DAM.RD3BSQ	. READ UNIT 3 (ERTS MSS DATA IN MOP BSQ FORMAT)
SPRT.SC DAM.RD3BSK	. READ UNIT 3 (DATA ON DISK IN PXBDEF FORMAT)
SPRT.SC DAM.RD3NUL	. READ UNIT 3 (SYNTHETIC DATA WHEN NO UNIT 3)
SPRT.SC DAM.READ7N	. READ DATA FROM DETECTION FILE(S) (UNITS 21-24)
SPRT.SC DAM.READ3	. READ UNIT 3 (ERTS MSS DATA)
SPRT.SC DAM.READ5	. FILL BUFFER FOR UNIT 5 (CARD READER OR TERMINAL)
SPRT.SC DAM.REG-NOM/LSAT-1-ERT	. NOMINAL REGISTRATION PARAMETERS
SPRT.SC DAM.REG-NOM/LSAT-2-ERT	. NOMINAL REGISTRATION PARAMETERS
SPRT.SC DAM.REG-NOM/LSAT-2-HOM	. NOMINAL REGISTRATION PARAMETERS
SPRT.SC DAM.REG-NOM/LSAT-2-SOM	. NOMINAL REGISTRATION PARAMETERS
SPRT.SC DAM.REG-NOM/LSAT-2-UTH	. NOMINAL REGISTRATION PARAMETERS
SPRT.SC DAM.REG-NOM/LSAT-3-ERT	. NOMINAL REGISTRATION PARAMETERS
SPRT.SC DAM.REG-NOM/LSAT-3-HOM	. NOMINAL REGISTRATION PARAMETERS
SPRT.SC DAM.REG-NOM/LSAT-3-SOM	. NOMINAL REGISTRATION PARAMETERS
SPRT.SC DAM.REG-NOM/LSAT-3-UTH	. NOMINAL REGISTRATION PARAMETERS
SPRT.SC DAM.REVERT	. REVERT EQUATIONS
SPRT.SC DAM.RITADD	. WRITE/ADD SPECIFIED SYMBOLIC ELEMENT
SPRT.SC DAM.RL2ISX	. CONVERT REAL TO SEXAGENARY ARRAY (INTEGER)
SPRT.SC DAM.RL2SX	. CONVERT REAL TO SEXAGENARY ARRAY (REAL)
SPRT.SC DAM.RL4SX	. COMPUTE REAL FROM SEXAGENARY ARRAY (REAL)
SPRT.SC DAM.ROTCHX	. ROTATE TWO MATRIX COLUMNS TO MAXIMIZE FUNCTION
SPRT.SC DAM.ROTCOL	. ROTATE TWO MATRIX COLUMNS
SPRT.SC DAM.ROTROW	. ROTATE TWO MATRIX ROWS
SPRT.SC DAM.R3TREC	. READ UNIT 3 (READ ONE RECORD FROM TAPE)
SPRT.SC DAM.SETMOD	. GET/SET MODE SWITCHES
SPRT.SC DAM.SHASAM	. SHARPEN SAMPLES IN PXBDEF FORMAT BUFFER
SPRT.SC DAM.SHFTBC	. SHIFT BITS CIRCULAR WITHIN WORDS OF ARRAY
SPRT.SC DAM.SPANS	. ENABLE/DISABLE SPANNING FOR UNIT 5
SPRT.SC DAM.SPLIT	. SPLIT REAL INTO SIGN.INTEGER.DECIMAL
SPRT.SC DAM.SREAD5	. SPANNED READ OF UNIT 5 (USED ONLY BY GET5)
SPRT.SC DAM.SSPR	. COMPUTE SUI'S & SUMS-OF-PRODUCTS
SPRT.SC DAM.STREG8	. STORE REGISTRATION PARAMETERS ON UNIT 8
SPRT.SC DAM.SUBWIN	. GENERATE SUBWINDOW MAPS
SPRT.SC DAM.TRACE/DAM	. TRACE CALLS TO FORTRAN ROUTINES
SPRT.SC DAM.TRECVR	. TAPE ERROR RECOVERY FOR UNIT 3 (MOP FORMAT TAPES)
SPRT.SC DAM.TSHAP3	. TAPE SHAP FOR UNIT 3 (MOP FORMAT TAPES)
SHSO.N .UNGET5	. BACK UP 1 DATA FIELD ON UNIT 5 INPUT (SEE GET5)
SPRT.SC DAM.VALKEY	. VALIDATE SECURITY KEY
SPRT.SC DAM.VARSON	. NORMALIZED VARIMAX ROTATION CRITERION
SPRT.SC DAM.VARSQU	. UN-NORMALIZED VARIMAX ROTATION CRITERION

**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**APPENDIX-N  
004**

SPRT.SC DAM.VERA4G  
SPRT.SC DAM.VERA4P  
SPRT.SC DAM.VERG4U  
SHSO.N .VIATO  
SPRT.SC DAM.WARN5  
SPRT.SC DAM.WINEXT  
SPRT.SC DAM.WININT  
SPRT.SC DAM.WRITE4  
SPRT.SC DAM.WRVERT  
SPRT.SC DAM.XREG77

- . WINDOW VERTICES: ADJUSTED MSS FOR GEOGRAPHIC
- . WINDOW VERTICES: ADJUSTED MSS FOR PRINT/PLOT
- . WINDOW VERTICES: GEOGRAPHIC FOR UTM
- . CALL 'VIA' 'TO' ROUTINES (SEE NVIATO)
- . PROCESS WARNING DIAGNOSTIC FOR UNIT 5
- . COMPUTE INTERCEPTS FOR WINDOW EXTERIOR
- . COMPUTE INTERCEPTS FOR WINDOW INTERIOR
- . WRITE COMMAND RECALL FILE (UNIT 4)
- . WRITE VERTEX COORDINATES FOR WINDOW
- . DUMP X-REGISTER IN OCTAL (FOR DEBUGGING)

**ORIGINAL PAGE IS  
OF POOR QUALITY**

SUBROUTINE ARE077( 8 DUMP SPECIFIED UNIVAC 1100 A-REGISTER IN OCTAL  
1 NUMREG) 8 NUMBER OF A-REGISTER (0 THRU 11)  
-----

HISTORY  
-----

E H SCHLOSSER LEC 12/05/79 ORIGINAL CODE

METHOD  
-----

ENCODE CONTENTS OF UNIVAC 1100 SERIES A-REGISTER IN OCTAL AND PRINT THEM.

THIS ROUTINE IS DESIGNED TO HELP DEBUG OBSCURE ERRORS IN THE  
COMPILER & OPERATING SYSTEM -- ON RETURN, ALL REGISTERS (EVEN THOSE IN  
THE VOLATILE 'MINOR REGISTER SET') ARE UNCHANGED.

MACHINE-DEPENDENT CODE  
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 6-BIT  
FIELDATA CHARACTERS. THE CALLING SEQUENCE IS THAT USED BY UNIVAC  
FORTRAN V.  
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.  
DIFFERENT COMPILERS (EG., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

EXTERNAL REFERENCES  
-----

ER PRINTS 8 UNIVAC X-8 EXEC REQUEST TO PRINT FIELDATA CHARACTER BUFFER

EXCEPTIONS  
-----

1. THE ROUTINE PRINTS A DIAGNOSTIC IF 0 <= NUMREG <= 15.

GLOBAL DECLARATIONS  
-----

NONE.

	AXRS	8 BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
\$100) . 1-BANK		
ARE077*	DS	A0.ABUF+0 .
	DS	A2.ABUF+2 .
	DS	A4.ABUF+4 .
	DS	A6.ABUF+6 .

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

ARE077  
002

	DS	A0.ABUF+0	.
	DS	A10.ABUF+10	.
	DS	A12.ABUF+12	.
	DS	A14.ABUF+14	.
.			
SUBNAME	LA	A3.1.X11	. A3 := W.B. WORD (W2 = ADDR OF SUBR NAME)
	LA	A1.0.A3	. A1 := SUBROUTINE NAME
	SA	A1.PBUF+0	. PUT SUBROUTINE NAME IN PRINT BUFFER
.			
REGNUM	LA	A0.*0.X11	. A0 := NUMREG
	JN	A0.BADNUM	. BAD IF NUMREG NEGATIVE
	T0.U	A0.16	. SKIP N1 IF 16 > NUMREG
	J	BADNUM	. (NUMREG) >= 16)
.			
	LA	A1.BIAS	. '0' IS NEGATIVE!
	LA	A2.BIAS	.
	LA	A3.ABUF.A0	. A3 := CONTENTS OF SPECIFIED A-REG
.			
UNPACK8	LOSL	A1.6	. SHIFT & PAD
	SSL	A2.3	. TOO FAR!
	LOSL	A2.3	. EXTRACT 3-BIT OCTAL DIGIT
	JN	A1.UNPACK8	. REPEAT TIL LEADING CHAR NOT '0'
.			
ENCODE8	AA	A1.BIAS	. CONVERT 6-BIT OCTAL
	AA	A2.BIAS	. TO FIELDATA
	DS	A1.PBUF+3	. AND PUT IN PRINT BUFFER
.			
ENCODE10	DSL	A0.36	. A1 := A0. A0 := 0
	DI.U	A0.10	. A0 := TENS DIGIT. A1 := UNITS DIGIT
	AA.U	A0.'000000'	. CONVERT TENS DIGIT TO FIELDATA CHAR
	AA.U	A1.'000000'	. CONVERT UNITS DIGIT TO FIELDATA CHAR
	SA.S2	A0.PBUF+2	. INSERT TENS CHARACTER IN BUFFER
	SA.S3	A1.PBUF+2	. INSERT UNITS CHARACTER IN BUFFER
.			
PRINT	PSRINT	(PF 1.5.PBUF)	. PRINT BUFFER
.			
RESTORE	DL	A0.ABUF+0	. RESTORE REGISTERS
	DL	A2.ABUF+2	.
	DL	A4.ABUF+4	.
	J	2.X11	. RETURN
.			
BADNUM	DL	A1.BADMSG	. PUT BAD MSG INTO ...
	DS	A1.PBUF+3	. ... PRINT BUFFER
	J	ENCODE10	. ENCODE BAD NUMREG & PRINT BUFFER
.			
\$(01) . D-BANK			
ABUF	RES	16	
PBUF	RES	1	. 1 WD FOR SUBROUTINE NAME
	RES	AREG 99	.
	RES	2	. 2 WDS FOR OCTAL ENCODING OF REGISTER
BADMSG		'(BAD NUMBER)'	.
PF	FORM	12.6.18	.
BIAS		'000000'	.
	END		

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

ARGRET  
001

SUBROUTINE ARGRET( 3 DETERMINE ACTUAL NUMBER OF ARGUMENTS & RETURN VECTOR  
0 NARGO. 3 NUMBER OF ACTUAL ARGUMENTS PASSED TO ROUTINE CALLING ARGRET  
0 KRETN) 3 RETURN K VECTOR FOR USE BY ROUTINE CALLING ARGRET  
-----

HISTORY  
-----

E H SCHLOSSER LEC 09/27/79 ORIGINAL CODE

METHOD  
-----

THIS ROUTINE ALLOWS THE INVOKING ROUTINE TO BE CALLED WITH A VARIABLE  
NUMBER OF ACTUAL ARGUMENTS. DETERMINE THAT NUMBER, AND RETURN PROPERLY.  
ARGRET SEARCHES THROUGH THE FORTRAN V ARGUMENT LIST PASSED TO THE  
INVOKING ROUTINE UNTIL IT FINDS THE FIRST VALID INSTRUCTION (SI<>0).  
IT ASSUMES THAT THIS INSTRUCTION IS PRECEDED BY THE FORTRAN V WALKBACK  
WORD AND THAT THE WALKBACK WORD IS PRECEDED BY THE ADDRESS OF THE LAST  
ACTUAL ARGUMENT.

MACHINE-DEPENDENT CODE  
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS. THE  
ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT, BUT THE  
METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT COMPILERS  
(EG.. UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

EXTERNAL REFERENCES  
-----

NONE.

EXCEPTIONS  
-----

VIOLATION OF THE FOLLOWING REQUIREMENTS MAY ABORT THE CURRENTLY EXECUTING  
PROGRAM:

1. THE INVOKING ROUTINE MUST DECLARE 2 EXTRA FORMAL ARGUMENTS, FOLLOWING  
THE LAST OPTIONAL ARGUMENT. THESE 2 ARGUMENTS MUST NEVER BE USED IN  
ANY WAY AT ALL!!
2. THE INVOKING ROUTINE MUST NOT ATTEMPT TO ACCESS ANY MISSING FORMAL  
ARGUMENT OR TO PASS IT AS AN ACTUAL ARGUMENT ON A CALL TO ANOTHER  
ROUTINE.
3. THE INVOKING ROUTINE MUST NOT DIRECTLY ACCESS ANY PRESENT FORMAL

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

ARGRET  
002

ARGUMENT WHICH IS OPTIONAL. (IT MAY BE ACCESSED INDIRECTLY. AS THE  
ACTUAL ARGUMENT ON A SUBROUTINE OR FUNCTION CALL.)

4. THE INVOKING ROUTINE MUST NOT PERFORM A NORMAL RETURN. BUT MUST  
INSTEAD PERFORM A RETURN KRETN.

GLOBAL DECLARATIONS

NONE.

	AXRS		BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
S(00) . 1-BANK			
ARGRET:	LA,H2	A0.2.X11	. ADDRESS OF W.B. PACKET
	LA,H2	A2.1.A0	. ADDR OF ADDR OF 0TH ARG ON PREV CALL
	LR,U	R1.30	. # OF ARG ADDRESSES TO SEARCH
	LX1,U	A2.1	. SEARCH INCREMENT IS 1
	SZ	A3	. 00 IS INVALID OPCODE
	SNE,S1	A3.0.*A2	. SEARCH FOR FIRST VALID OPCODE
	NOP		. NOT FOUND: INVOKER WILL CRASH ON RETURN
	LX1,U	A2.0	. ZERO INCREMENT PORTION
	ANA,H2	A2.1.A0	. A2 := RETURN K VECTOR
	AU,XU	A2.-2	. A3 := ACTUAL NUMBER OF ARGUMENTS
	SA	A2.*1.X11	. KRETN := A2
	SA	A3.*0.X11	. NARGS := A3
	J	3.X11	. RETURN TO INVOKER
	END		

```

.   LMJ  A0.ATRACE  .  CONDITIONAL TRACE OF CALL TO ASSEMBLER ROUTINE
.   '???????'  .  NAME OF ROUTINE
.   -----
.
.
.
.   HISTORY
.   -----
.
.   E H SCHLOSSER  LEC  06/18/74  ORIGINAL CODE
.   E H SCHLOSSER  LEC  01/07/80  KOMXQT MACRO
.
.
.   METHOD
.   -----
.
.   IF MTRACE IN KOMXQT LABELLED COMMON IS NOT ZERO, THEN PRINT THE NAME
.   OF THE CALLING ROUTINE FROM THE ARGUMENT.
.
.
.   MACHINE-DEPENDENT CODE
.   -----
.
.   WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 8-BIT
.   FIELDATA CHARACTERS.
.   THE METHOD OF PASSING ARGUMENTS IS **NOT** THAT USED BY UNIVAC FORTRAN V.
.   IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES AND
.   DIFFERENT MACHINES.
.
.
.   EXTERNAL REFERENCES
.   -----
.
.   ER PRINTS
.
.
.   EXCEPTIONS
.   -----
.
.   1. THIS ROUTINE MUST NOT BE CALLED BY A FORTRAN ROUTINE!!
.
.
.   GLOBAL DECLARATIONS
.   -----
.
.           AXRS          .  STANDARD UNIVAC 1100 REGISTER MNEMONICS
.           KOMXQT        .  COMMON PROGRAM EXECUTION SWITCHES
.
.
.   LOCAL DECLARATIONS
.   -----
.
.   S(00) . 0-BANK
.   A0X1      0.0.0.0.1.1 . SPACE 1 LINE. PRINT 1 WORD
.
.
.   PROCEDURE

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

ATRACE/DAM  
002

. -----

S(01) . 1-BANK

ATRACE\*

TNZ

J

LXI

ER

J

END

MTRACE

1.A0

A0.AOXI

PRINTS

1.A0

. CHECK TRACE SWITCH

. SWITCH IS OFF -- RETURN

. XI = PRINT SPECS

. PRINT 1 WORD

. RETURN



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

BOX-CHW  
001

```

000 111 111 000 AAA 0000
CCC 0000 EEEEE FFFFF 000 H H 111 J
K K L M M N N 000 PPPP QQQ RRRR
SSS TTTT U U V V W W X X Y Y ZZZZ
)
. . . ( XX 777 1
00 1 222 333 4 5555 66 77777
000 999 11
0 0 1 1 000 A A B B
C C D D E F 0 0 H H I J
K K L M M N N N O O P P Q Q R R
S S T U U V V W W X X Y Y Z
)
. . . ( XX 11 7 7 1 \
0 0 11 2 2 3 3 44 5 6 7
0 0 9 9 11 /
0 0 1 1 0000 A A B B
C C D D E F 0 H H I J
K K L M M N N N O O P P Q Q R R
S S T U U V V W W X X Y Y Z
)
. . . ( 11 7 1 \
0 0 1 2 3 4 4 5555 6 7
0 0 9 9 /
0 0 1 1 000 AAAAA 000
C C D D EEE FFF 0 HHHHH I J
K K L M M N N N O O PPPP Q Q RRRR
SSS T U U V V W W W X Y Z
)
. . . ( 11 7 1 \
0 0 1 222 33 4 4 5 6666 7
000 9999 11 /
0 0 1 1 0000 A A B B
C C D D E F 0 00 H H I J
K K L M M N N O O P Q Q Q R R
S S T U U V V W W W X X Y Z
)
. . . ( 11 7 1 \
0 0 1 2 3 4444 5 6 6 7
0 0 9 9 11 /
0 0 1 1 000 A A B B
C C D D E F 0 0 H H I J
K K L M M N N O O P Q Q R R
S S T U U V V W W W X X Y Z
)
. . . ( 11 7 1 \
0 0 1 2 3 3 4 5 5 6 6 7
0 0 9 9 11 /
0000 111 111 000 A A 0000
CCC 0000 EEEEE F 0000 4 H 111 JJJ
K K LLLL M M N N 000 000 Q Q R R
SSS T UUU V W W X X Y ZZZZ
)
. . . ( XX 7 1 .
00 111 22222 333 4 555 666 7
000 99 11

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

BYTDMP  
001

```

SUBROUTINE BYTDMP( 8 EXTRACT/SCALE/DUMP BYTE FIELDS (2'S COMPL BINARY)
.
1 ISPEC, 8 SPECIFICATION ARRAY OF 8 WORDS PER BYTE FIELD:
C      1 STARTING BYTE OF FIELD
C      2 LENGTH OF SUB-FIELD IN BYTES (-1 = END SPECS)
C      3 INTEGER BASE OF SCALING FACTOR
C      4 INTEGER EXPONENT OF SCALING FACTOR
C      556 NAME OF FIELD (7 TO 12 CHARACTERS)
1 (BUFR) 8 BUFFER CONTAINING FIELDS OF INTERNAL BYTES
-----
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      01/28/78      ORIGINAL CODE
C      C A HELMKE        LEC      10/15/79      CALLS TO NEW GETBYT/DBY/QBY. 14KTWO
C
C METHOD
C -----
C
C      EXTRACT FIELD. CONVERT TO INTEGER FROM TWOS COMPLEMENT BINARY, SCALE,
C      AND PRINT.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      ASSUMES 8 CHARACTERS PER WORD.
C      OCTAL FORMAT SPEC.
C
C EXTERNAL REFERENCES
C -----
C
C      GETBYT      8 GET NON-NEGATIVE INTEGER FROM BYTE IN BYTE STRING
C      GETDBY      8 GET NON-NEGATIVE INTEGER FROM DOUBLE BYTE IN BYTE STRING
C      GETQBY      8 GET INTEGER FROM QUADRUPLE BYTE IN BYTE STRING
C      INTEGER 14KTWO 8 INTEGER FOR TWOS COMPLEMENT BINARY
C
C EXCEPTIONS
C -----
C
C      1. ANY SUB-FIELDS LONGER THAN 4 BYTES ARE IGNORED.
C
C      2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C

```

C LOCAL DECLARATIONS

C -----  
C

```

      INTEGER ISPEC(1)      & ARGUMENT
      INTEGER IBUF(1)      & ARGUMENT
      INTEGER IDYNAM(2)    & NAME OF SUB-FIELDS IN CURRENT FIELD
      INTEGER ISP          & POINTER TO FIRST SPEC FOR CURRENT FIELD
      INTEGER IBYT         & STARTING BYTE OF CURRENT SUB-FIELD
      INTEGER LENBYT       & LENGTH IN BYTES OF SUB-FIELDS IN CURRENT FIELD
      INTEGER LENBIT       & LENGTH IN BITS ....
      INTEGER KTHFLD       & CURRENT SUB-FIELD CONTENTS (PARTIAL WORD, THOS COMPL)
      INTEGER INTFLD       & CURRENT SUB-FIELD CONTENTS (WHOLE WORD, INTEGER)
      INTEGER INTMP        & TEMPORARY
      REAL SCALEF          & SCALE FACTOR FOR ALL SUB-FIELDS IN CURRENT FIELD
      REAL RADFLD          & SCALED VALUE OF CURRENT SUB-FIELD (NORMALLY RADIAN)
      REAL DEOFLD          & RADFLD CONVERTED TO DEGREES

```

C

C

C PROCEDURE

C -----

C

C

C WRITE HEADING

C

```

      WRITE(6,115)
115 FORMAT(
      & ' STRT L   NAME                OCTAL      INTEGER      RADIAN     ' ,
      & 'DEGREES' )

```

C

C

C INITIALIZE SPEC POINTER

C

```

      ISP=1

```

C

C

C GET SPECS FOR SUB-FIELDS IN CURRENT FIELD

C

```

200 IBYT=ISPEC(ISP)
    LENBYT=ISPEC(ISP+1)
    LENBIT=LENBYT*8
    IF(LENBYT.LT.1) GO TO 900      & END OF SPECS/BUFFER
    IF(LENBYT.GT.4) GO TO 600      & SKIP FIELDS WITH SUB-FIELDS LONGER THAN 4
    SCALEF=FLOAT(ISPEC(ISP+2))*ISPEC(ISP+3)
    IDYNAM(1)=ISPEC(ISP+4)
    IDYNAM(2)=ISPEC(ISP+5)

```

C

C

C EXTRACT SUB-FIELD

C

```

300 GO TO (320,340,360,380),LENBYT

```

C

```

320      CALL OCTOBYT(KTHFLD, IBUF,IBYT)      & 1 BYTE SUB-FIELD
      GO TO 400

```

C

```

340      CALL OCTOBYT(KTHFLD, IBUF,IBYT)      & 2 BYTE SUB-FIELD
      GO TO 400

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

BYTOMP  
003

```

C
300      CALL OCTOBYT(INTERP, 1BUFR,1BYT)  & 3 BYTE SUB-FIELD
        CALL OCTOBYT(KTHFLD, 1BUFR,1BYT+2)
        KTHFLD=KTHFLD+INTERP*2+16
        GO TO 400

C
300      CALL OCTOBYT(KTHFLD, 1BUFR,1BYT)  & 4 BYTE SUB-FIELD
C
C
C CONVERT SUB-FIELD TO SIGNED INTEGER. SCALE IT. PRINT IT
C
400      INTFLD=1+KTHO(KTHFLD,LENBYT)      & CONVERT TO INTEGER FROM THOS CONPL
        RADFLD=FLOAT(INTFLD)*SCALEF
        DEOFLO=RADFLD*180./3.14159265
        WRITE(6,515) 1BYT,LENBYT,1BYNAM,KTHFLD,INTFLD,RADFLD,DEOFLO
515      FORMAT(1X,14,1X,11,1X,2A6,1X,011,1X,110,1X,F14.6,1X,F10.5)

C
C
C PREPARE FOR NEXT SUB-FIELD IN CURRENT FIELD
C
1BYT=1BYT+LENBYT
IF(1BYT.LT.1SPEC(1SP+6)) GO TO 300      & SUB-FIELD EXISTS. DO IT!

C
C
C PREPARE FOR NEXT FIELD
C
600      1SP=1SP+6
        GO TO 200

C
C
C END OF SPECS/BUFFER
C
900      RETURN
        END

```

SUBROUTINE CALCHA 8 CALIBRATE BUFFER ASSIGNMENTS FOR RAW CHANNELS

```

C
C
C
C HISTORY
C -----
C
C      C A HELMKE      LEC      12/20/79      REQUIREMENTS
C      C A HELMKE      LEC      12/21/79      ALGORITHM DESIGN
C      C A HELMKE      LEC      12/21/79      ALGORITHM CODING
C      J C CRISP      LENS CO  02/19/80      BUF FOR RAW CHAN ONLY IF PRESENT & USED
C      E M SCHLOSSER  LENS CO  05/11/80      POLAR: RAW CHAN 1 IN BUFFER 1
C
C
C
C METHOD
C -----
C
C      RAW CHANNELS:
C      DETERMINE RAW CHANNELS USED AS THOSE DEFINED AS LIMIT CHANNELS
C      IN THE LIMCH ARRAY.
C      DETERMINE RAW CHANNELS PRESENT AS THOSE WITH A NON-BLANK BAND
C      IN THE NERBAN ARRAY.
C      FOR ALL RAW CHANNELS PRESENT & USED. UPDATE NBFCHR ARRAY TO
C      ASSIGN BUFFER NUMBER 1 TO RAW CHANNEL 1.
C
C      LINEAR & POLAR TRANSFORMED CHANNELS:
C      DETERMINE RAW CHANNELS USED AS THOSE WITH NON-ZERO RADIANCE
C      TRANSFORMATION LINEAR WEIGHTS IN RTLWOT ARRAY.
C      DETERMINE RAW CHANNELS PRESENT AS THOSE WITH A NON-BLANK BAND
C      IN THE NERBAN ARRAY.
C      FOR ALL RAW CHANNELS PRESENT & USED. UPDATE NBFCHR ARRAY TO
C      ASSIGN BUFFER NUMBER 1 TO RAW CHANNEL 1.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      MDWARN      8 PRINT/LOG/COUNT 'WARNING' MESSAGES
C      DOUBLE PRECISION COS4IN  8 VAR-LENGTH (8<=CHAR) STRING FOR INT STRING
C      DOUBLE PRECISION COS4CS  8 VAR-LENGTH (8<=CHAR) STRING FOR CHR STRING
C
C
C EXCEPTIONS
C -----
C
C      1. IF REQUIRED CHANNEL IS NOT PRESENT ISSUE MDWARN.
C
C      2. IF CHANNEL TYPE INVALID ISSUE MDWARN.
C

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

CALCHA  
002

```

C
C GLOBAL DECLARATIONS
C -----
C
      INCLUDE KONXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES,COUNTERS
      INCLUDE KONIRT.LIST      & COMMON IRRADIANCE TRANSFORMATION COEFFICIENTS
      INCLUDE KONKLS.LIST      & COMMON CLASSIFICATION INFO
      INCLUDE KONNER.LIST      & COMMON ERTS SCENE PARAMETERS

C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER NBF              & BUFFER NUMBER
C      INTEGER NCHAN            & CHANNEL NUMBER (NO INPUT FILE ASSIGNED)
C      INTEGER ICHRAW            & RAW CHANNEL NUMBER
C      INTEGER ICHLIN            & LINEAR CHANNEL NUMBER

C
C
C PROCEDURE
C -----
C
C      CALL TRACE

C
C
C CHECK TO SEE IF INPUT FILE IS ASSIGNED
C
      DO 100 ICHRAW=1,NERCHA
        IF (NERBAN(ICHRAW).NE.' ') GO TO 190 & IF NOT BLANK,ASSUME INPUT FILE
      100 CONTINUE

C
C
C IF INPUT FILE NOT ASSIGNED, SET BUFFER POINTERS
C
      IF (IRTTYP.NE.'RAW') GO TO 190
      DO 120 NBF=1,NLINCH
        NCHAN=LINCH(NBF)
        IF (NCHAN.GT.5) GO TO 120
        NBFCHR(NCHAN)=NBF
      120 CONTINUE
      GO TO 900 & INPUT FILE NOT ASSIGNED
      190 DO 160 ICHRAW=1,NERCHA
        NBFCHR(ICHRAW)=ICHRAW
      160 CONTINUE
      GO TO 900

C
C
C CHANNEL TYPE RAW
C
      190 IF (IRTTYP.NE.'RAW') GO TO 300
      IF (NLINCH.LE.5) GO TO 900
      DO 200 NBF=1,NLINCH
        ICHRAW=LINCH(NBF)
        IF (ICHRAW.GT.5) GO TO 200
        IF (NERBAN(ICHRAW).EQ.' ') CALL MDWARN (

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CALCHA  
003

```

      * 'RAW CHANNEL'.CBS4IN(ICHRAW,3),' NOT PRESENT')
      IF(NERBAN(ICHRAW).EQ.' ') GO TO 200
      NBFCHR(ICHRAW)=NBF      3 RAW CHANNEL PRESENT
200 CONTINUE
      GO TO 900
C
C
C CHANNEL TYPE LINEAR
C
300 IF(IRTYP.NE.'LIN') GO TO 400
      DO 330 NBF=1,NLINC
          ICHLIN=LINC:NBF
          DO 310 ICHRAW=1,NERCHA
              IF (RTLWGT(ICHRAW,ICHLIN).EQ.0.0) GO TO 310
              IF (NERBAN(ICHRAW).EQ.' ') CALL MDWARN (
                  * 'RAW CHANNEL'.CBS4IN(ICHRAW,3),' FOR LINEAR CHANNEL'.
                  CBS4IN(ICHLIN,3),' NOT PRESENT')
              IF(NERBAN(ICHRAW).EQ.' ') GO TO 330
              NBFCHR(ICHRAW)=ICHRAW      3 RAW CHANNEL PRESENT
310 CONTINUE
330 CONTINUE
      GO TO 900
C
C
C CHANNEL TYPE POLAR
C
400 IF(IRTYP.NE.'POL') GO TO 800
      DO 430 ICHRAW=1,NERCHA
          IF(RTLWGT(ICHRAW,1).EQ.0.0).AND.(RTLWGT(ICHRAW,2).EQ.0.0)
              GO TO 430
          IF(NERBAN(ICHRAW).EQ.' ') CALL MDWARN(
              * 'RAW CHANNEL'.CBS4IN(ICHRAW,3),' NOT PRESENT FOR POLAR ')
          IF(NERBAN(ICHRAW).EQ.' ') GO TO 430
          NBFCHR(ICHRAW)=ICHRAW      3 RAW CHANNEL PRESENT
430 CONTINUE
      GO TO 900
C
C
C INVALID CHANNEL TYPE
C
800 CALL MDWARN(
    * 'BAD CHANNEL TYPE ---','.CBS4CS(IRTYP,(1),6))
C
C
C
900 RETURN
C
      END

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CALCOL  
001

```
      SUBROUTINE CALCOL  & CALIBRATE COLOR/INTENSITY IN SYMBOL TABLE
      -----
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEMSCO  02/22/80      STUBBED
C
C
C METHOD
C -----
C
C      NOT YET IMPLEMENTED.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      ASSUMES AT LEAST 8 CHARACTERS PER KSYM ELEMENT.
C
C
C EXTERNAL REFERENCES
C -----
C
C      GETICE      & GET INTEGER-CHARACTER-EQUIVALENT FROM CHARACTER STRING
C
C
C EXCEPTIONS
C -----
C
C      NONE.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMSYM.LIST      & COMMON SYMBOL TABLE
C
C
C LOCAL DECLARATIONS
C -----
C
C      NONE.
C
C
C PROCEDURE
C -----
C
C      CALL TRACE
C
C
C      RETURN
C      END
```



SUBROUTINE CALSCA: 3 CALIBRATE PPD TRANSFORMATION COEFFICIENTS FOR SCALE

1 PLINCH. 3 PPD LINES PER INCH  
1 PCINCH1 3 PPD COLUMNS PER INCH

-----

C

C

C

C HISTORY

C -----

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

E M SCHLOSSER LEC 03/02/74 ORIGINAL CODE  
E M SCHLOSSER LEC 01/30/79 MAX SCALE 1/1000000 & FIX D0170

C METHOD

C -----

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

CALIBRATE THE MAPPING EQUATIONS FOR SCALE. OUTPUT DEVICE. AND OFFSET.  
TYPICALLY. FOR A STANDARD LINE-PRINTER PLINCH=6.0 AND PCINCH=10.0. AND  
FOR A PEN PLOTTER CALIBRATED IN INCHES BOTH PLINCH AND PCINCH WOULD  
BE 1.0.

C MACHINE-DEPENDENT CODE

C -----

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

NONE.

C EXTERNAL REFERENCES

C -----

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

MDWARN 3 PRINT/COUNT/LOG 'WARNING' DIAGNOSTIC MESSAGE  
REVERT 3 COMPUTE INVERSE TRANSFORMATION COEFFICIENTS

C EXCEPTIONS

C -----

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

1. INVALID SCALE GENERATES A WARNING DIAGNOSTIC.

C GLOBAL DECLARATIONS

C -----

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

INCLUDE KOMNER.LIST 3 COMMON ERTS SCENE PARAMETERS  
INCLUDE KOMFIT.LIST 3 COMMON ADJUSTMENT/REGISTRATION PARAMETERS  
INCLUDE TRFORM.LIST 3 DEFINE COORDINATE TRANSFORMATION FUNCTIONS

C LOCAL DECLARATIONS

C -----

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

REAL RFO 3 REPRESENTATIVE FRACTION DENOMINATOR

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CALSCA  
002

```

C
C PROCEDURL
C -----
C
C
C      CALL TRACE
C
C
C CHECK SCALE
C
      IRFD=1ABS(IRFD)      8 FLAG SCALE AS CALIBRATED (POSITIVE)
      RFD=IRFD
      IF((IRFD.GE.20000).AND.(IRFD.LE.1000000)) GO TO 150
      CALL MOWARN('SCALE NOT DEFINED')
      RFD=1.

C
C
C COMPUTE LINEAR COEFFICIENTS IN FORWARD PRINTER/PLOTTER EQUATIONS
C      PPDLIN=CORPPD(1)*CORLIN+CORPPD(2)*CORSAM+CORPPD(3)
C      PPDCOL=CORPPD(4)*CORLIN+CORPPD(5)*CORSAM+CORPPD(6)
C
      150 CORPPD(1)=CORSRM(1)*39.37*PLINCH/RFD
      CORPPD(2)=CORSRM(2)*39.37*PLINCH/RFD
      CORPPD(4)=CORSRM(4)*39.37*PCINCH/RFD
      CORPPD(5)=CORSRM(5)*39.37*PCINCH/RFD

C
C
C FIND MINIMUM PPD LINE & COLUMN WITHIN SCENE
C
      PPDLMN=-9E+31
      PPDCHN=-9E+31
      ADJSAM=1.
      CORSAM=CORS4A(1.,ADJSAM)
      DO 170 MSALIN=0,NERLIN,NERLIN
      CORLIN=MSALIN      8 FUTURE CORRECTION
      PPDLIN=CORPPD(1)*CORLIN+CORPPD(2)*CORSAM
      PPDCOL=CORPPD(4)*CORLIN+CORPPD(5)*CORSAM
      PPDCHN=AMINI(PPDCHN,PPDCOL)
      PPDLMN=AMINI(PPDLMN,PPDLIN)
      170 CONTINUE

C
C
C BIAS PPD EQUATIONS SO PPD COORDINATES ARE POSITIVE WITHIN 300 UNITS OF SCENE
C
      CORPPD(3)=300.0-PPDLMN      8 PPDLIN > +300 WITHIN SCENE
      CORPPD(6)=300.0-PPDCHN      8 PPDCOL > +300 WITHIN SCENE

C
C
C COMPUTE INVERSE PRINTER/PLOTTER COEFFICIENTS
C      CORLIN=PPDCOR(1)*PPDLIN+PPDCOR(2)*PPDCOL+PPDCOR(3)
C      CORSAM=PPDCOR(4)*PPDLIN+PPDCOR(5)*PPDCOL+PPDCOR(6)
C
      CALL REVERT(CORPPD,PPDCOR)

C
C
      RETURN

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**CALSCA  
003**

**END**

**CALSPA**  
**001**

```

C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      01/06/74      ORIGINAL CODE
C      E M SCHLOSSER      LEC      02/13/79      EXPAND DOCUMENTATION
C      D A BECK            LEC      09/21/79      SENSITIZE TO ORIGIN
C
C
C METHOD
C -----
C
C      CALIBRATE THE DISPLAY EQUATIONS FOR SPACING, AND OFFSET, AND ORIGIN.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      MDWARN      & PRINT/COUNT/LOG 'WARNING' DIAGNOSTIC MESSAGE
C      REVERT      & COMPUTE INVERSE TRANSFORMATION COEFFICIENTS
C
C
C EXCEPTIONS
C -----
C
C      1. INVALID SPACING GENERATES A WARNING DIAGNOSTIC.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMNER.LIST      & COMMON ERTS SCENE PARAMETERS
C      INCLUDE KOMFIT.LIST      & COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C      INCLUDE TRFORM.LIST      & DEFINE COORDINATE TRANSFORMATIONS
C      INCLUDE KOMQHW.LIST      & COMMON OUTPUT WINDOW PACKETS
C      INCLUDE WINDEF.LIST      & DEFINE STRUCTURE OF WINDOW PACKETS
C
C
C LOCAL DECLARATIONS
C -----
C
C      REAL PPDLO      & FPD LINE OF ORIGIN
C      REAL PPOCO      & FPD COLUMN OF ORIGIN
C
C
C PROCEDURE
C -----

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CALSPA  
002

```

C
C
C      CALL TRACE
C
C
C CHECK SPACING
C
      IF((MSAOWH(WLIN, WSP100)).NE.0).AND.
&      (MSAOWH(WSAM, WSP100).NE.0)) GO TO 150
          CALL MDWARN( 'SPACING NOT DEFINED')
          GO TO 900
C
C
C COMPUTE LINEAR COEFFICIENTS IN FORWARD PRINTER/PLOTTER EQUATIONS
C      PPDLIN=CORPPD(1)*CORLIN+CORPPD(2)*CORSAM+CORPPD(3)
C      PPDCOL=CORPPD(4)*CORLIN+CORPPD(5)*CORSAM+CORPPD(6)
C
      150 CORPPD(1)=100./FLOAT(MSAOWH(WLIN, WSP100))
          CORPPD(2)=0.
          CORPPD(4)=0.
          CORPPD(5)=100./FLOAT(MSAOWH(WSAM, WSP100))
C
C
C CALIBRATE CONSTANT TERMS IN FORWARD PRINTER/PLOTTER EQUATIONS
C
      CORPPD(3)=300.      & > +300 WITHIN SCENE
      CORPPD(6)=300.      & > +300 WITHIN SCENE
C
C
C COMPUTE PPD COORDINATES OF ORIGIN BASED ON SPACING
C
      PPDL0=PPDL4C(FLOAT(MSAOWH(WLIN, WOR10)),FLOAT(MSAOWH(WSAM, WOR10)))
      PPDC0=PPDC4C(FLOAT(MSAOWH(WLIN, WOR10)),FLOAT(MSAOWH(WSAM, WOR10)))
C
C
C REFINE CONSTANT TERMS IN FORWARD PRINTER/PLOTTER EQUATIONS
C
      CORPPD(3)=CORPPD(3)-(PPDL0-A1N1(PPDL0))
      CORPPD(6)=CORPPD(6)-(PPDC0-A1N1(PPDC0))
C
C
C COMPUTE INVERSE PRINTER/PLOTTER COEFFICIENTS
C      CORLIN=PPDCOR(1)*PPDLIN+PPDCOR(2)*PPDCOL+PPDCOR(3)
C      CORSAM=PPDCOR(4)*PPDLIN+PPDCOR(5)*PPDCOL+PPDCOR(6)
C
      CALL REVERT(CORPPD,PPDCOR)
C
C
C ADD SMALL BIAS VALUE TO CONSTANT TERMS IN CORPPD AND PPDCOR FOR CORRECTION
C
      CORPPD(3)=CORPPD(3)+0.0001
      CORPPD(6)=CORPPD(6)+0.0001
C
      PPDCOR(3)=PPDCOR(3)+0.0001
      PPDCOR(6)=PPDCOR(6)+0.0001
C

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**CALSPA  
003**

**C  
C RETURN TO CALLING ROUTINE  
C  
900 RETURN  
END**

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

CALSYN  
001

SUBROUTINE CALSYN    & CALIBRATE SYMBOL TABLE FOR PRINTING

```

C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      12/11/78      ORIGINAL CODE
C      E M SCHLOSSER      LEC      12/20/79      REPLACE KSYBIT WITH NCISYM
C
C
C METHOD
C -----
C
C      CHANGE ALL UNDEFINED SYMBOLS TO '?' AND DETERMINE MAXIMUM LENGTH IN
C      CHARACTERS OF SYMBOLS.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      MOVCSY      & MOVE CHARACTER STRING
C      INTEGER LENCST      & LENGTH OF CHARACTER STRING
C
C
C EXCEPTIONS
C -----
C
C      NONE.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KMSYH.LIST      & COMMON SYMBOL TABLE
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER NSYM      & SYMBOL NUMBER
C      INTEGER KSY      & CURRENT SYMBOL (4 CHARS), LEFT-ALIGNED & PADDED W/ ZERO
C
C
C PROCEDURE
C -----
C
C      CALL TRACE
C
C
C      NCISYM=0

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CALSYN  
002

```
DO 300 NSYM=1, KSYMSZ
  KSY=0
  CALL MOVCSY(KSY, (1), (4), KSY(NSYM), (1), (4), ' ')
  IF(KSY.EQ.0) CALL MOVCSY(KSY(NSYM), (1), (4),
    '7', (1), (4), ' ')
  NCISYM=MAX0(NCISYM, LENCST(KSY(NSYM), (4)))
300 CONTINUE
C
C
  RETURN
END
```



DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CALWIN  
001

SUBROUTINE CALWIN: 8 CALIBRATE ENVELOPES OF OUTPUT WINDOW PACKETS

1 EXPAND: 8 PPD UNITS TO EXPAND PPD WINDOW BY AT ALL EDGES

C  
C  
C  
C HISTORY  
C -----

C	E M SCHLOSSER	LEC	10/05/73	ORIGINAL CODE
C	E M SCHLOSSER	LEC	10/29/79	UPGRADE DOCUMENTATION

C  
C  
C METHOD  
C -----

C COMPUTE PRINTER/PLOTTER ORIGIN. COMPUTE OUTPUT WINDOW ENVELOPE, RELATIVE  
C TO THE ORIGIN, IN THE COORDINATE SYSTEM IN WHICH THE WINDOW VERTICES  
C ARE DEFINED. CONVERT ENVELOPE TO ABSOLUTE COORDINATES BY ALGEBRAICALLY  
C ADDING THE ORIGIN. COMPUTE PPD ENVELOPE AROUND THIS ENVELOPE JUST  
C DERIVED FROM THE VERTICES. COMPUTE THE GEOGRAPHIC, UTM, AND SCANNER  
C ENVELOPES AROUND THE PPD ENVELOPE. IF THE WINDOW VERTICES WERE NOT  
C IN SCANNER COORDINATES, TRANSFORM THEM TO SCANNER COORDINATES.

C  
C MACHINE-DEPENDENT CODE  
C -----

C NONE.

C  
C EXTERNAL REFERENCES  
C -----

C	MDWARN	8 PRINT/COUNT/LOG 'WARNING' DIAGNOSTIC MESSAGE
C	INWIN	8 COMPUTE INTEGER ENVELOPE AROUND INTEGER VERTICES
C	ENVWIN	8 COMPUTE REAL ENVELOPE AROUND REAL VERTICES
C	INVORI	8 ALGEBRAICALLY ADD INTEGER ORIGIN TO INTEGER ENVELOPE
C	ENVORI	8 ALGEBRAICALLY ADD REAL ORIGIN TO REAL ENVELOPE
C	P4A	8 PRINT/PLOT DEVICE COORDINATES FOR ADJUSTED MSS COORDINATES
C	Q4P	8 GEOGRAPHIC COORDINATES FOR PRINT/PLOT DEVICE COORDINATES
C	A4P	8 ADJUSTED MSS COORDINATES FOR PRINT/PLOT DEVICE COORDINATES
C	A4O	8 ADJUSTED MSS COORDINATES FOR GEOGRAPHIC COORDINATES
C	U4O	8 UTM COORDINATES FOR GEOGRAPHIC COORDINATES
C	Q4U	8 GEOGRAPHIC COORDINATES FOR UTM COORDINATES
C	VER4O	8 ADJUSTED MSS VERTICES FOR GEOGRAPHIC VERTICES
C	VER4U	8 GEOGRAPHIC VERTICES FOR UTM VERTICES
C	VER4P	8 ADJUSTED MSS VERTICES FOR PRINT/PLOT DEVICE VERTICES
C	DNPHN	8 DUMP WINDOW PACKETS

C  
C EXCEPTIONS  
C -----

C 1. IF NEITHER CALSPA NOR CALSCA HAVE BEEN CALLED PREVIOUSLY, THEN THE  
C RESULTS OF CALWIN ARE UNDEFINED.

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CALWIN  
002

```

C
C      2. IF NO COORDINATE SYSTEM IS DEFINED FOR THE ORIGIN OR FOR THE WINDOW
C      VERTICES. 'WARNING' DIAGNOSTIC(S) ARE GENERATED.
C
C      3. IF THE WINDOW VERTICES ARE IN UTM COORDINATES BUT NO UTM ZONE OR
C      CENTRAL MERIDIAN IS DEFINED. A 'WARNING' DIAGNOSTIC IS GENERATED.
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
C      INCLUDE KCMFIT.LIST      & COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C      INCLUDE WINDEF.LIST      & DEFINE STRUCTURE OF WINDOW PACKETS
C      INCLUDE KONIWH.LIST      & COMMON INPUT WINDOW PACKETS
C      INCLUDE KOMOWH.LIST      & COMMON OUTPUT WINDOW PACKETS
C      INCLUDE MAXINT.LIST      & DEFINE MAXIMUM INTEGER
C
C
C LOCAL DECLARATIONS
C -----
C
C      LOGICAL CALO4U          & TRUE IF CALLING O4U. FALSE IF NOT
C      INTEGER NDSAVE          & TEMPORARY SAVE LOCATION FOR VALUE OF NOTOTL ON ENTRY
C      INTEGER NOD1,NOD2       & NODE SUBSCRIPTS FOR ENVELOPES
C      REAL PPDLIN,PPDCOL      & PPD LINE, COLUMN OF ENVELOPE CORNERS
C      REAL ADJLIN,ADJSAM      & ADJUSTED LINE, SAMPLE OF ENVELOPE CORNERS
C
C
C PROCEDURE
C -----
C
C      CALL TRACE
C
C MARK ENVELOPE AS UNDEFINED
C
C      NDSAVE=NDTOTL
C      KSYOWH(MMIN)='NUL'
C      KSYOWH(MMAX)='NUL'
C
C
C CHECK ORIGIN & TRANSFORM INTO PPD COORDINATES
C
C      IF(KSYOWH(WORIG).EQ.'88888') KSYOWH(WORIG)='NUL'      & TEMP PATCH
C      IF(KSYOWH(WORIG).NE.'NUL') GO TO 100
C      CALL MDWARN('NO ORIGIN')
C      GO TO 900
C 100 CALL P4A(PPDOWH(WLIN,WORIG),PPDOWH(WCOL,WORIG),
C      -      FLOAT(MSAOWH(WLIN,WORIG)),FLOAT(MSAOWH(WSAM,WORIG)))
C
C
C CASE ON WINDOW TYPE
C
C      IF(INDFATL.NE.0) GO TO 900
C      IF(KSYOWH(MVER).EQ.'SCA') GO TO 300

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

CALWIN  
003

```

      IF((KSYOMH(MVER).EQ.'DEG').OR.
&      (KSYOMH(MVER).EQ.'MIN')) GO TO 400
      IF((KSYOMH(MVER).EQ.'KM ').OR.
&      (KSYOMH(MVER).EQ.'MET')) GO TO 600
      IF(KSYOMH(MVER).EQ.'PRI') GO TO 700
      CALL MDHARN( 'NO WINDOW')
      GO TO 900

```

C  
C  
C SCANNER (LINE-LENGTH ADJUSTED) COORDINATES  
C

```

300 CALL INVMIN(MSAOMH)
   CALL INVORI(MSAOMH)
   CALL ENVP4A
   CALL ENVA4P
   CALL ENV04P
   CALL ENVU4P
   GO TO 900

```

C  
C  
C GEOGRAPHIC COORDINATES  
C

```

400 CALL ENVMIN(GEDOMH)
   CALL ENVORI(GEDOMH)
   CALL ENVP40
   CALL ENVA4P
   CALL ENVU4P
   CALL VERA40
   GO TO 900

```

C  
C  
C UTM COORDINATES  
C

```

600 IF((UTMCHD.NE.0).AND.(UTMCHD.LE.180.)) GO TO 630
   CALL MDHARN( 'NO UTM ZONE')
   GO TO 900
630 CALL ENVMIN(UTMOMH)
   CALL ENVORI(UTMOMH)
   CALL ENVP4U
   CALL ENVA4P
   CALL ENV04P
   CALL VER04U
   CALL VERA40
   GO TO 900

```

C  
C  
C PRINT/PLOT DEVICE COORDINATES  
C

```

700 CALL ENVMIN(PPDOMH)
   CALL ENVORI(PPDOMH)
   CALL ENVP4P
   CALL ENVA4P
   CALL ENV04P
   CALL ENVU4P
   CALL VERA4P

```

C

**CALWIN**  
**004**

**N-31**

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CALWIN  
005

```

C
400 PPDOHW(WLIN.WMIN)=+9E+35
   PPDOHW(WLIN.WMAX)=-9E+35
   PPDOHW(WSAM.WMIN)=+9E+35
   PPDOHW(WSAM.WMAX)=-9E+35
C
   DO 460 NOD1=WMIN.WMAX
     DO 430 NOD2=WMIN.WMAX
       IF(CALQ4U) CALL Q4U(
         &      GEDOWH(WLAT.NOD1).GEDOWH(WLON.NOD2),
         &      UTHOWH(WEA.NOD1).UTHOWH(WNO.NOD2).UTCHMD)
       CALL A4G(ADJLIN.ADJSAM,
         &      GEDOWH(WLAT.NOD1).GEDOWH(WLON.NOD2))
       ADJLIN=AMINI(ADJLIN.FLOAT(MSAIHW(WLIN.WMAX)+200)) & WITHIN 200
       ADJLIN=AMAXI(ADJLIN.FLOAT(MSAIHW(WLIN.WMIN)-200)) & LINES AND
       ADJSAM=AMINI(ADJSAM.FLOAT(MSAIHW(WSAM.WMAX)+200)) & SAMPLES
       ADJSAM=AMAXI(ADJSAM.FLOAT(MSAIHW(WSAM.WMIN)-200)) & OF SCENE
       CALL P4A(PPDOLIN.PPDCOL,ADJLIN.ADJSAM)
       PPDOHW(WLIN.WMIN)=AMINI(PPDOHW(WLIN.WMIN).PPDOLIN)
       PPDOHW(WLIN.WMAX)=AMAXI(PPDOHW(WLIN.WMAX).PPDOLIN)
       PPDOHW(WSAM.WMIN)=AMINI(PPDOHW(WSAM.WMIN).PPDCOL)
       PPDOHW(WSAM.WMAX)=AMAXI(PPDOHW(WSAM.WMAX).PPDCOL)
430      CONTINUE
460 CONTINUE
C
C
C
C EXPAND PRINT/PLOT ENVELOPE ON ALL SIDES
C
   PPDOHW(WLIN.WMIN)=PPDOHW(WLIN.WMIN)-PXPAND
   PPDOHW(WSAM.WMIN)=PPDOHW(WSAM.WMIN)-PXPAND
   PPDOHW(WLIN.WMAX)=PPDOHW(WLIN.WMAX)+PXPAND
   PPDOHW(WSAM.WMAX)=PPDOHW(WSAM.WMAX)+PXPAND
C
C
C
   ENTRY ENVP4P & COMPUTE PPDOHW ENVELOPE AROUND PPDOHW ENVELOPE
C
C
C INSURE PRINT/PLOT COORDINATES ARE ALWAYS POSITIVE
C
810 PPDOHW(WLIN.WMIN)=AMAXI(PPDOHW(WLIN.WMIN),2.)
   PPDOHW(WCOL.WMIN)=AMAXI(PPDOHW(WCOL.WMIN),2.)
C
C
   RETURN
C
C
C
C
C
C
   INTERNAL
   SUBROUTINE ENVA4P & COMPUTE MSAOHV ENVELOPE AROUND PPDOHW ENVELOPE
C
   MSAOHV(WLIN.WMIN)=+MAXINT
   MSAOHV(WLIN.WMAX)=-MAXINT

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CALMIN  
006

```

      MSAOWH(WSAM,WMIN)=+MAXINT
      MSAOWH(WSAM,WMAX)=-MAXINT
C
      DO 200 NOD1=WMIN,WMAX
        DO 100 NOD2=WMIN,WMAX
          CALL A4P(ADJLIN,ADJSAM,
-             PPDOHW(WLIN,NOD1),PPDOHW(WCOL,NOD2))
          MSAOWH(WLIN,WMIN)=MIN0(MSAOWH(WLIN,WMIN),IFIX(ADJLIN))
          MSAOWH(WLIN,WMAX)=MAX0(MSAOWH(WLIN,WMAX),IFIX(ADJLIN))
          MSAOWH(WSAM,WMIN)=MIN0(MSAOWH(WSAM,WMIN),IFIX(ADJSAM))
          MSAOWH(WSAM,WMAX)=MAX0(MSAOWH(WSAM,WMAX),IFIX(ADJSAM))
100      CONTINUE
200      CONTINUE
C
      RETURN
C
C
C
C
C
C
      INTERNAL
      SUBROUTINE ENVG4P 3 COMPUTE GEDOWH ENVELOPE AROUND PPDOHW ENVELOPE
C
      GEDOWH(WLAT,WMIN)=+9E+35
      GEDOWH(WLAT,WMAX)=-9E+35
      GEDOWH(WLON,WMIN)=+9E+35
      GEDOWH(WLON,WMAX)=-9E+35
C
      DO 200 NOD1=WMIN,WMAX
        DO 100 NOD2=WMIN,WMAX
          CALL G4P(GEDLAT,GEDLON,
-             PPOOHW(WLIN,NOD1),PPDOHW(WCOL,NOD2))
          GEDOWH(WLAT,WMIN)=AMIN1(GEDOWH(WLAT,WMIN),GEDLAT)
          GEDOWH(WLAT,WMAX)=AMAX1(GEDOWH(WLAT,WMAX),GEDLAT)
          GEDOWH(WLON,WMIN)=AMIN1(GEDOWH(WLON,WMIN),GEDLON)
          GEDOWH(WLON,WMAX)=AMAX1(GEDOWH(WLON,WMAX),GEDLON)
100      CONTINUE
200      CONTINUE
C
      RETURN
C
C
C
C
C
C
      INTERNAL
      SUBROUTINE ENVU4P 3 COMPUTE UTMOWH ENVELOPE AROUND PPDOHW ENVELOPE
C
      UTMOWH(WEA,WMIN)=+9E+35
      UTMOWH(WEA,WMAX)=-9E+35
      UTMOWH(WNO,WMIN)=+9E+35
      UTMOWH(WNO,WMAX)=-9E+35
C
      DO 200 NOD1=WMIN,WMAX

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

CALWIN  
007

```
      DO 100 NOD2=WMIN,WMAX
        CALL G4P(GEDLAT,GEDLON,
                PPODWH(MLIN,NOD1),PPODWH(MCOL,NOD2))
        CALL U40(UTHE,UTHN, GEDLAT,GEDLON,UTHCHO)
        UTHOWH(WEA,WMIN)=AMIN1(UTHOWH(WEA,WMIN),UTHE)
        UTHOWH(WEA,WMAX)=AMAX1(UTHOWH(WEA,WMAX),UTHE)
        UTHOWH(WNO,WMIN)=AMIN1(UTHOWH(WNO,WMIN),UTHN)
        UTHOWH(WNO,WMAX)=AMAX1(UTHOWH(WNO,WMAX),UTHN)
100    CONTINUE
200    CONTINUE
      RETURN
C
C      END
```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

CLOSE3  
001

```

SUBROUTINE CLOSE3      & CLOSE INPUT MSS/RBV FILE ASSIGNED TO UNIT 3
-----
C
C
C HISTORY
C -----
C
C      MARY TOMPKINS      LEC      08/20/79      REQUIREMENTS
C      J C CRISP          LEC      09/25/79      ALGORITHM DESIGN
C      J C CRISP          LEC      09/25/79      ALGORITHM CODING
C
C METHOD
C -----
C
C      CHECK LU3SEQ(1). IF BIP CALL CL3BIP. ELSE RETURN.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      CL3BIP      & CLOSE AND VERIFY EOF ON ERTS TAPE
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES. COUNTERS
C      INCLUDE KOHLU3.LIST      & PACKET/POINTERS FOR UNIT 3
C
C LOCAL DECLARATIONS
C -----
C
C      NONE
C
C PROCEDURE
C -----
C
C      CALL TRACE
C
C
C      IF (LU3SEQ(1).EQ.'BIP') CALL CL3BIP
C
C
C 900 RETURN
C
C      END

```



DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CLOSE4  
001

```

SUBROUTINE CLOSE4( 3 WRITE EOF ON UNIT 4 (SDF COMMAND RECALL FILE)
1 NUMCRD) 3 CARD IMAGE NUMBER AFTER WHICH TO WRITE END-OF-FILE
-----
C
C
C
C      E H SCHLOSSER      LEC      03/17/75      ORIGINAL CODE
C      E H SCHLOSSER      LEC      11/09/79      CHECK NUMCRD RANGE
C
C
C METHOD
C -----
C
C      WRITE UNIVAC EXEC-8 SDF EOF CONTROL WD TO SECTOR 4 NUMCRD+1.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      VERRY!!
C
C
C EXTERNAL REFERENCES
C -----
C
C      ERION      3 INITIATE I/O AND WAIT FOR COMPLETION
C
C
C EXCEPTIONS
C -----
C
C      1. IF NUMCRD < 2, THE IMAGE IS NOT WRITTEN TO THE RECALL FILE.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMIO.LIST      3 COMMON I/O FUNCTIONS
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER LU4PKT(8)      3 I/O PACKET
C      DATA (LU4PKT(1),1-1,2) /'RECALL'      ' '      3 FILE NAME
C      INTEGER KWEOf /077777777777/ 3 SDF CONTROL WD: END-OF-FILE
C
C
C PROCEDURE
C -----
C
C      CALL TRACE
C
C
C CHECK NUMCRD
C
C      IF(NUMCRD.LT.2) GO TO 900

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CLOSEN  
002

```
C
C
C INITIALIZE PACKET & WRITE SDF END-OF-FILE CONTROL WORD
C
  IOSIZE(LU4PKT)=1
  IOADDR(LU4PKT)=LOC(KHEOF)
  IOSECT(LU4PKT)=NUMCRD+1
  IOFUNC(LU4PKT)='SC'      & WRITE
  CALL ER10W(LU4PKT)
C
C
C COMMON RETURN
C
  900 RETURN
  END
```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

CLOSPR  
001

```

SUBROUTINE CLOSPR  & CLOSE ALTERNATE PRINT FILE(S)
-----
C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      07/02/73      ORIGINAL CODE
C      MARY TOMPKINS      LEC      01/05/80      USE NEW CHARACTER ROUTINES
C
C
C METHOD
C -----
C
C      DETERMINE IF ALTERNATE PRINT FILES EXIST. IF FILE EXIST FOR:
C      FILE TO TAPE
C          BRKPT AND ISSUE NOTE ACKNOWLEDGING BRKPT TO TAPE.
C      FILE TO DISK
C          REQUEST EXEC TO PRINT FILES ONTO DISK. RESTORE
C          INSTALLATION DEFAULTS. DRAIN BUFFER. WRITE EOF.
C          AND QUEUE FOR PRINTING.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      USES EXEC-8 CONTROL STATEMENTS
C
C
C EXTERNAL REFERENCES
C -----
C
C      ERFITH      & RETRIEVE FACILITIES ASSIGNMENT INFORMATION
C      MONOTE      & PRINT/LOG/COUNT 'NOTE' MESSAGES
C      ERCSF      & SUBMIT EXEC-8 CONTROL STATEMENT FUNCTION
C      PRINC      & SET PRINT LINES PER INCH
C      PRMRO      & SET PRINT MARGINS & LINES
C      CST4IN      & CHARACTER STRING FOR INTEGER
C      MOVCSF      & MOVE CHARACTER STRING
C      INTEGER ICE  & INTEGER CHAR EQUIVALENT (OF FIRST CHR IN STRING)
C
C
C EXCEPTIONS
C -----
C
C      NONE.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOHXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES. COUNTERS
C      INCLUDE KOHALT.LIST      & COMMON ALTERNATE PRINT FILE COUNTERS. POINTERS
C
C LOCAL DECLARATIONS
C -----

```

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

CLOSPR  
002

```

C      INTEGER IDFILE(13)      & 13 WORD PKT. USED TO OBTAIN UPDATE FILE INFO.
C      DATA (IDFILE(N),N=1,2)/'10','/' & FILE NAME
C
C      INTEGER
C CHAR      000000000111111111222222222333333333444444444
C      1234567890123456789012345678901234567890123456789
C      4 JBRKPT(4)/'BRKPT      IN .      '/' & WRITE END-OF-FILE
C      4 JFREE(4)/'FREE      IN .      '/' & FREE & CATALOG
C      4 JFREE1(4)/'FREE.1    IN .      '/' & FREE & INHIBIT CATALOGING
C      5 JSYM(5) /'SYM      *DAMPRT-N..1.MNEMON . '/' & QUEUE TO PRINT 1 TIMES
C
C
C PROCEDURE
C -----
C
C      CALL TRACE('CLOSPR')
C
C
C SPECIAL HANDLING FOR ALTERNATE PRINT FILE ON TAPE
C
C      IF(MALTH.LE.0) GO TO 900      & NO ALTERNATE PRINT FILES
C      CALL ERFITH(IDFILE)
C      IF(ICE(IDFILE(7)),LE.0) GO TO 900 & NO ALT PRT FILES
C      IF(ICE(IDFILE(7)),GT.15) GO TO 300 & NOT TAPE
C      CALL ERCSF(NAO,'BRKPT 10 . ')
C      CALL MONOTE('PRINTER OUTPUT ON TAPE:')
C      WRITE(6,125) IDFILE(12), (IDFILE(N),N=3,11)
125  FORMAT(6X,'REEL '.A6,6X,11A6)
C      GO TO 900
C
C
C REQUEST EXEC TO PRINT ALT PRINT FILES ON DISK
C
C      300 NMAX=MALTH-1
C      CALL MOVCS('JSYM,22,6. MNEMON,1,6.' ')
C      DO 600 N = 0,NMAX
C      NUNIT=10*N
C      IF(INDFATL.NE.0) WRITE(NUNIT,305)
305  FORMAT('0*** FATAL ERROR *** ')
C      WRITE(NUNIT,315) N
315  FORMAT('1'/'0'/'0'/'0'/'
C      1          19(' . . .')/19(' . . .')/
C      2          9(' . . .'),'END '.11.' . 9(' . . .')/
C      3          19(' . . .')/19(' . . .')/'0')
C
C
C SPECIAL PRINT CONTROL FUNCTIONS TO RESTORE INSTALLATION DEFAULTS
C
C      CALL PRINC(NUNIT,LSINCH,KONPRT)
C      IF(LPAGE.NE.LSPAGE) CALL PRTHRO(NUNIT,LSPAGE,0,0)
C
C
C DRAIN BUFFER, WRITE EOF, QUEUE FOR PRINTING

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CLOSPR  
003

```

C
      CALL CST4IN(JBRKPT,17,1, N,1)
      CALL ERCSF(NAO,JBRKPT)
      CALL CST4IN(JFREE0,17,1, N,1)
      CALL ERCSF(NAO,JFREE0)
      IF(NCOPY.LT.1) GO TO 500
          CALL CST4IN(JSYM,17,1, N,1)
          CALL CST4IN(JSYM,20,1, NCOPY,1)
          CALL ERCSF(NAO,JSYM)
          GO TO 600
500  WRITE(6,550) N
550  FORMAT(' ***USER MUST 8SYM OR 8DELETE FILE *DAMPRT-*,11,*,')
600  CONTINUE
      GO TO 900

C
C
C
C
C
C
      ENTRY DLETPR 8 DELETE ALTERNATE PRINT FILES
      -----
C
C
C
      CALL TRACE('DLETPR')
      CALL ERFITM(IDFILE)
      IF(ICE(IDFILE(7)),LE,0) GO TO 900 8 NO ALT PRT FILES
          NMAX=MALTM-1
          DO 700 N=0,NMAX
              CALL CST4IN(JBRKPT,17,1, N,1)
              CALL ERCSF(NAO,JBRKPT)
              CALL CST4IN(JFREE1,17,1, N,1)
              CALL ERCSF(NAO,JFREE1)
700  CONTINUE
C
C
C NORMAL RETURN
C
C 900 RETURN
C
      END

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CLSHDD  
001

SUBROUTINE CLSHDD: 8 PRINT ID/CLASSIFICATION HEADING

1 NUNIT) 8 LOGICAL UNIT NUMBER TO WRITE ON

```

C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      07/02/73      ORIGINAL CODE
C      E H SCHLOSSER      LEC      06/29/78      IDENTIFY TOLERANCE
C      E H SCHLOSSER      LEC      01/20/79      IDENTIFY DET FILE AS RAD/DEN/CLA
C      J C CRISP           LEC      01/04/80      UPGRADE DOCUMENTATION
C      J C CRISP           LEMSCO   05/16/80      IDENTIFY # OF CHANNELS IN DET FILE
C
C
C
C METHOD
C -----
C
C      THIS SUBROUTINE PRINTS COMMON ID/CLASSIFICATION HEADING ON NUNIT.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      FORMAT SPECIFICATIONS ASSUME 8 CHARACTERS PER INTEGER
C
C
C EXTERNAL REFERENCES
C -----
C
C      NONE.
C
C
C EXCEPTIONS
C -----
C
C      NONE.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KONXQT.LIST      8 COMMON PROGRAM EXECUTION SWITCHES. COUNTERS
C      INCLUDE KONNER.LIST      8 COMMON ERTS SCENE PARAMETERS
C      INCLUDE KOMKLS.LIST      8 COMMON CLASSIFICATION INFO
C      INCLUDE WINDEF.LIST      8 DEFINE STRUCTURE OF WINDOW PACKETS
C      INCLUDE KOMDET.LIST      8 COMMON DETECTION FILE WINDOW PACKETS AND DATES
C
C
C
C LOCAL DECLARATIONS
C -----
C
C      NONE.
C
C
C

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CLSHDO  
002

C PROCEDURE

C -----

C

CALL TRACE

C

C

IF((NUNIT.NE.6).AND.(MIDATAC.NE.0)) GO TO 900      8 DATA/CHECKOUT MODE

C

C

```

WRITE(NUNIT,150) NERTS.NERDAY.NERMON.NERYR.NERSEL.NERSAZ
150  FORMAT('0'/'0'/
1 6X.'ERTS SCENE: '.11.J4.'-'J5/
2 6X.'DATE: '.12.1X.A3.1X.12/
3 6X.'SUN ELEV: '.12.' DEGREES'/
4 6X.'SUN AZIMUTH: '.13.' DEGREES')

```

C

C

```

WRITE(NUNIT,450)
6 (JENMOY(N).N=1.4).
7 (MSADHW(HLIN.WMIN.N).N=1.4).
8 (MSADHW(HLIN.WMAX.N).N=1.4).
9 (MSADHW(HSAM.WMIN.N).N=1.4).
A (MSADHW(HSAM.WMAX.N).N=1.4)
450  FORMAT(1X./
5 6X.'CCT STRIPS:'.11X.'1'.6X.'2'.6X.'3'.6X.'4'/
6 6X.'DATE CLASSIFIED:'.4(1X.A6)/
7 6X.'MINIMUM LINE: '.4I7/
8 6X.'MAXIMUM LINE: '.4I7/
9 6X.'MINIMUM SAMPLE: '.4I7/
A 6X.'MAXIMUM SAMPLE: '.4I7)

```

C

C

```

WRITE(NUNIT,475)
B (MTERAL(1).1=1.4).KLSTYP.NERCHA.
C LCVTOL
475  FORMAT(
B 6X.'MATERIAL(S) DETECTED: '.4A8.' ('.A3.'.'11.'1'/
C 6X.'TOLERANCE: '.117)

```

C

C

900 RETURN  
END

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CL3BIP  
001

SUBROUTINE CL3BIP 3 CLOSE AND VERIFY EOF ON ERTS BIP TAPE

```

C
C
C
C HISTORY
C -----
C
C      J C CRISP      LEC      08/20/79      REQUIREMENTS
C      C A HELMKE     LEC      11/13/79      ALGORITHM DESIGN
C      C A HELMKE     LEC      11/18/79      ALGORITHM CODING
C
C
C METHOD
C -----
C
C      READ TAPE UNTIL END-OF-FILE.  IF NCCT <> 4 RETURN.  ELSE READ
C      SIAT LOGICAL TAPE HEADER RECORD.  CHECK LU3BFM.  IF '00'. CALL
C      BST400.  CALL CST400.  EXTRACT AND WRITE SIAT NUMBER AND
C      PROCESSING DATE.  LOCATE AND READ RBV COMPUTATIONAL DATA
C      RECORD.  CHECK LU3BFM.  IF '00'. CALL BST400.  CALL CST400.
C      EXTRACT/DECODE CENTER/NADIR/HEADING AND COMPUTE/VERIFY/UPDATE
C      PITCH AND ROLL.  IF MDWARN <> 0 OR MDUMP <> 0. CALL BYTOMP.
C      ELSE READ SIAT MSS COMPUTATIONAL DATA RECORD.  CHECK LU3BFM.
C      IF '00' CALL BST400.  CALL CST400.  EXTRACT/DECODE/VERIFY/
C      UPDATE PITCH AND ROLL.  IF MDWARN <> 0 OR MDUMP <> 0. CALL
C      BYTOMP.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      BST400      3 INTERNAL BYTE STRING FOR 8-BIT BYTE STRING
C      CST400      2 CHARACTER STRING FOR EBCDIC BYTE STRING
C      GET0BY      3 GET NON-NEGATIVE INTEGER FROM QUAD BYTE IN BYTE STRING
C      GETDBY      3 GET NON-NEGATIVE INTEGER FROM DOUBLE BYTE IN BYTE STRING
C      ER10W      3 INITIATE AND WAIT FOR COMPLETION ON I/O
C      BYTOMP      3 EXTRACT/SCALE/DUMP BYTE FIELDS
C      I4KTH0      3 SIGNED INTEGER FROM TWO'S-COMPLEMENT BINARY
C      PITROL      3 ESTIMATE PITCH AND ROLL
C      M0FATL      3 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C      MDWARN      3 PRINT/LOG/COUNT 'WARNING' MESSAGES
C      MDNOTE      3 PRINT/LOG 'NOTE' MESSAGES
C      MDCLR0      3 CLEAR 'WARNING' COUNT
C      COS4CS      3 VARIABLE-LENGTH STRING FOR FIXED-LENGTH STRING
C      COS4IN      3 VARIABLE-LENGTH STRING FOR INTEGER
C
C      DOUBLE PRECISION COS4CS
C      DOUBLE PRECISION COS4IN
C
C

```



DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CL301P  
002

C EXCEPTIONS

C -----

C

C 1. THE FOLLOWING CONDITIONS GENERATE THE FOLLOWING ERRORS WHILE  
C VERIFYING EOF OF IMAGE RECORDS:

C NO END OF FILE ON ERTS TAPE	FATAL
C EOF OUT OF POSITION	FATAL
C LOST POSITION	FATAL
C BAD FILE	FATAL

C 2. THE FOLLOWING CONDITIONS GENERATE THE FOLLOWING ERRORS WHILE  
C READING COMPUTATIONAL DATA RECORDS:

C BAD PITCH FROM CONTROL NET	WARN
C BAD ROLL FROM CONTROL NET	WARN
C SIAT CENTER/NADIR INCONSISTENT WITH PITCH	WARN
C SIAT CENTER/NADIR INCONSISTENT WITH ROLL	WARN
C LOST POSITION	WARN
C BAD RECORD	WARN
C BAD FILE	WARN

C GLOBAL DECLARATIONS

C -----

C

INCLUDE KOMXQT.LIST	% COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
INCLUDE KOMLU3.LIST	% I/O AND UNPACKING DATA FOR MSS/RBV DATA (UNIT 3)
INCLUDE KOMIO.LIST	% FORTRAN MANIPULATION OF ASSEMBLER I/O PACKETS
INCLUDE KOMNER.LIST	% ERTS SCENE PARAMETERS
INCLUDE NULCST.LIST	% DEFINE NULL CHARACTER STRING

C

C

C LOCAL DECLARATIONS

C -----

C

INTEGER L3BUFR(81)	% BUFFER FOR LOGICAL UNIT 3
DATA L3BUFR(1)/0/	% TO ELIMINATE SPURIOUS COMPILER DIAGNOSTIC

C

C SPECS FOR FIELDS IN SIAT RBV COMP DATA RECORD

START	LEN	SCALE	NAME
C INTEGER IORBV(74)/			
% 017.	02.	01.-01.	'ALT CHANGE'.
% 019.	10.	00.-00.	'*ZSCDIC*'
% 037.	04.	10.-06.	'CENTER LAT'.
% 041.	04.	10.-06.	'CENTER LON'.
% 045.	04.	10.-06.	'NADIR LAT'.
% 049.	04.	10.-06.	'NADIR LON'.
% 053.	04.	01.-01.	'ALT METERS'.
% 057.	04.	10.-03.	'GMT SEC'.
% 061.	04.	10.-06.	'HEADING'.
% 065.	04.	10.-06.	'PITCH'.
% 069.	04.	10.-06.	'ROLL'.
% 073.	04.	10.-06.	'YAW'.
% 077.	-1/		

C

C DIMENSION IOMSS(134)

C SPECS FOR FIELDS IN SIAT MSS COMP DATA RECORD

DAH PACKAGE APPENDIX N  
UTILITY ROUTINES

CL3BIP  
603

```

C      START LEN  SCALE  NAME
      DATA (10MSS(N),N=1,66)/
      & 017. 02. 01.+01. 'ALT CHANGE'.
      & 039. 02. 32.-01. 'ALT N MILES'.
      & 093. 02. 02.-17. 'ROLL & CTR'.
      & 095. 02. 02.-17. 'PITCH & CTR'.
      & 097. 02. 02.-17. 'YAW & CTR'.
      & 099. 02. 02.-17. 'ROLL (9)'.
      & 077. 02. 02.-17. 'PITCH (9)'.
      & 099. 02. 02.-17. 'YAW (9)'.
      & 113. 02. 02.-17. 'IMAGE SKEW'.
      & 119. 02. 02.-17. 'VELOC CHANGE'.
      & 117. 04. 10.-06. 'MEAN PITCH'/
      DATA (10MSS(N),N=67,134)/
      & 121. 04. 10.-06. 'MEAN ROLL'.
      & 125. 04. 10.-06. 'MEAN YAW'.
      & 129. 04. 10.-06. 'PITCH RATE'.
      & 133. 04. 10.-06. 'ROLL RATE'.
      & 137. 04. 10.-06. 'YAW RATE'.
      & 141. 04. 01.+01. 'MEAN ALT'.
      & 145. 04. 01.+01. 'ALT RATE'.
      & 149. 04. 10.-03. 'GMT SEC(11)'.
      & 193. 04. 10.-06. 'NADIRLAT(11)'.
      & 237. 04. 10.-06. 'NADIRLON(11)'.
      & 281. 04. 01.+01. 'ALTMETER(11)'.
      & 325.-1/
      REAL DEGRAD /57.29577951/      & DEGREES PER RADIAN
      INTEGER IIPAST      & RECORD # OF RECORD 10 BEYOND LAST RECORD ON VOLUME
      INTEGER IATREC      & STAT RECORD NUMBER (RBV)
      REAL S37CLA      & CENTER LATITUDE (RBV)
      REAL S41CLO      & CENTER LONGITUDE (RBV)
      REAL S45NLA      & NADIR LATITUDE (RBV)
      REAL S49NLO      & NADIR LONGITUDE (RBV)
      REAL S81HMY      & HEADING MINUS YAW (RBV)
      REAL V85PIT      & PITCH (RBV)
      REAL V89ROL      & ROLL (RBV)
      REAL S93ROL      & ROLL (MSS)
      REAL S95PIT      & PITCH (MSS)
      REAL S87ROL      & ROLL (MSS)
      REAL S89PIT      & PITCH (MSS)

C
C
C PROCEDURE
C -----
C
C      CALL TRACE
C
C
C
C      100 IF(INDFATL.NE.0)GO TO 900
C          IF(LU3SEQ(1).NE.'BIP')GO TO 900      & NOT A BIP TAPE
C
C
C VERIFY END-OF-FILE
C
C      WRITE(6,219)

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

CL3BIP  
004

```

215 FORMAT('0.. VERIFYING EOF ON ERTS BIP TAPE')
IRPAST=2340+10  & SET 10 RECORDS PAST LAST RECORD
C
C
C READ IMAGE RECORDS ON TAPE UNTIL END-OF-FILE
C
DO 350 IIRREAD=LU3RBF,IRPAST
CALL R3TREC(L3BUFR,81,ITSTAT, 1,200,300,80)
IF(ITSTAT.EQ.' ' .OR. ITSTAT.EQ.'BADR') GO TO 330
IIRTEND=IIRREAD
IF(IIRTEND.EQ.2340) GO TO 400  & EOF AT CORRECT POSITION
CALL M0FATL(C8S4CS(ITSTAT,(1),4),' AFTER LINE',
& C8S4IN(IIRTEND,6),'DISREGARD OUTPUT')
IF(ITSTAT.EQ.'EOF') GO TO 400  & EOF AT INCORRECT POSITION
IF(ITSTAT.EQ.'BADF')GO TO 900
330 IF(MOD(IIRREAD,256).EQ.0)WRITE(6,335)
335 FORMAT(' ..')  & SECURITY BLANKET
350 CONTINUE
CALL M0FATL(
& 'NO END OF FILE ON BIP TAPE--DISREGARD ALL OUTPUT')
GO TO 900
C
C
C READ SIAT LOGICAL TAPE HEADER RECORD (#1)
C
400 IF(NCCT.NE.4)GO TO 900
IOSIZE(LU3PKT)=81
IOADDR(LU3PKT)=LOC(L3BUFR)
IOWAIT(LUEPKT)=10
IATREC=1
IOFUNC(LU3PKT)='3K'  & READ FORWARD
CALL ER10W(LU3PKT)
ISTAT=IOCODE(LU3PKT)
IF(ISTAT.NE.' ')GO TO 870
C
C
C CONVERT SIAT NUMBER AND PROCESSING DATE FROM EBCDIC TO FIELDATA
C AND WRITE OUT
C
IF(LU3BFH.EQ.'88')CALL BST4B8(L3BUFR, L3BUFR,24)
CALL CST4E8(L3BUFR,L3BUFR,24)
WRITE(6,405)(L3BUFR(N),N=1,4)
405 FORMAT(
& '0'.8('.....')/
& '0 SIAT NUMBER & DATE OF TAPE PREPARATION: '/
& '0'.4A8/
& '0'.8('.....')//)
C
C
C READ SIAT RBV COMPUTATIONAL DATA RECORD (#5 OR #8)
C
410 CONTINUE
IATREC=IATREC+1
IOFUNC(LU3PKT)='3K'
CALL ER10W(LU3PKT)
ISTAT=IOCODE(LU3PKT)

```

DAH PACKAGE APPENDIX H  
UTILITY ROUTINES

CL3B1P  
005

```

      IF(ISTAT.NE.' ') GO TO 870
      IF(IATREC.GT.8) GO TO 870      & NO RBV RECORD
      NBYTES=NB4NI(IONWOS(LU3PKT))
      IF(LU3BFH.EQ.'88') NBYTES=NBYTES*9/8
      IF(NBYTES.GT.86) GO TO 410      & TOO LONG--NOT RBV RECORD--TRY AGAIN

```

C  
C  
C  
C

EXTRACT CENTER/NAOIR/HEADING & COMPUTE/VERIFY/UPDATE PITCH & ROLL

```

      IF(LU3BFH.EQ.'88') CALL BST488(L3BUFR.L3BUFR.76)
      CALL GETQBY(ITEMP.L3BUFR.(37))
      S37CLA=14KTWO(ITEMP.32)*10.**-06*DEGRAD
      CALL GETQBY(ITEMP.L3BUFR.(41))
      S41CLO=-14KTWO(ITEMP.32)*10.**-6*DEGRAD
      CALL GETQBY(ITEMP.L3BUFR.(45))
      S45NLA=14KTWO(ITEMP.32)*10.**-6*DEGRAD
      CALL GETQBY(ITEMP.L3BUFR.(49))
      S49NLO=-14KTWO(ITEMP.32)*10.**-6*DEGRAD
      CALL GETQBY(ITEMP.L3BUFR.(61))
      S61HMY=14KTWO(ITEMP.32)*10.**-6*DEGRAD
      CALL PITROL (
      & S37CLA.S41CLO.S45NLA.S49NLO.S61HMY.ALTKM.EPITD.EROLD)
      IF(PITDEG.LT.-99.) PITDEG=EPITD
      IF(ROLDEG.LT.-99.) ROLDEG=EROLD
      IF(ABS(PITDEG-EPITD).GT.0.15) CALL MDWARN(
      = 'BAD PITCH FROM CONTROL NET')
      IF(ABS(ROLDEG-EROLD).GT.0.15) CALL MDWARN(
      = 'BAD ROLL FROM CONTROL NET')

```

C  
C  
C  
C

EXTRACT & SCALE & DUMP FIELDS FROM RBV COMP DATA RECORD

```

      IF(MBATCH.EQ.0) CALL MDCLRW( NULCST)      & IGNORE WARNING IF NOT BATCH
      IF( (MDWARN.EQ.0) .AND. (MDUMP.EQ.0) ) GO TO 500
      WRITE (8,435) IATREC
435   FORMAT('1*** SIAT RECORD '.11.'--RBV COMPUTATIONAL DATA ***')
      CALL BYTDMPI(ORBV.L3BUFR)
      CALL GETQBY(ITEMP.L3BUFR.(65))
      V65PIT=14KTWO(ITEMP.32)*10.**-6*DEGRAD
      CALL GETQBY(ITEMP.L3BUFR.(69))
      V69ROL=14KTWO(ITEMP.32)*10.**-6*DEGRAD
      500 CONTINUE

```

C  
C  
C  
C

READ SIAT MSS COMPUTATIONAL DATA RECORD (\*6 OR \*7)

```

      IATREC=IATREC+1
      IOFUNC(LU3PKT)='3K'
      CALL ER10W(LU3PKT)
      ISTAT=IOCODE(LU3PKT)
      IF(ISTAT.NE.' ') GO TO 870      & TAPE READ ERROR

```

C  
C  
C  
C

EXTRACT/VERIFY/UPDATE PITCH & ROLL FROM MSS DATA RECORD

```

      IF(LU3BFH.EQ.'88') CALL BST488(L3BUFR.L3BUFR.324)

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

CL3B1P  
006

```

CALL MOCLRW( NULCST)      3 CLEAR WARNINGS
CALL GETDBY(ITEMP.L38UFR.(53))
S53ROL=14KTWO(ITEMP.16)*2.00-17*DEGRAD
CALL GETDBY(ITEMP.L38UFR.(55))
S55PIT=14KTWO(ITEMP.16)*2.00-17*DEGRAD
CALL GETDBY(ITEMP.L38UFR.(67))
S67ROL=14KTWO(ITEMP.16)*2.00-17*DEGRAD
CALL GETDBY(ITEMP.L38UFR.(85))
S85PIT=14KTWO(ITEMP.16)*2.00-17*DEGRAD
IF(PITDEG.LT.-99) PITDEG=(S55PIT+S85PIT)/2.
IF(ROLDEG.LT.-99) ROLDEG=(S53ROL+S67ROL)/2.
IF(ABS(PITDEG-S55PIT).GT.0.20) CALL MDWARN(
  * 'SIAT CENTER/NADIR INCONSISTENT WITH PITCH')
IF(ABS(ROLDEG-S53ROL).GT.0.20) CALL MDWARN(
  * 'SIAT CENTER/NADIR INCONSISTENT WITH ROLL')
IF(MBATCH.EQ.0) CALL MOCLRW( NULCST)      3 IGNORE WARNING IF NOT BATCH
IF( (MDWARN.EQ.C) .AND. (MDUMP.EQ.0) ) GO TO 900

C
C
C DUMP MSS COMPUTATIONAL DATA RECORD
C
WRITE(6,515)1ATREC
515 FORMAT('1000 SIAT RECORD '.11,'--MSS COMPUTATIONAL DATA ***')
CALL BYDMP(10MSS.L38UFR)
GO TO 900

C
C
C CHECK AND FLAG TAPE ERRORS
C
870 IF(1STAT.EQ.'EOF') GO TO 880
CALL MDWARN(CBS4CS(1STAT.(1),4))
GO TO 900
880 CALL MDNOTE('NO SIAT FILE')

C
C
C USER MUST 3FREE, 3REWIND, OR 3LOCATE TAPE
C
900 RETURN
END

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

CROPOW  
001

SUBROUTINE CROPOW: 8 CROP OUTPUT WINDOW TO FIT INPUT WINDOW & PPD HARDWARE

1 NPRLMX. 8 MAXIMUM NUMBER OF PRINT LINES ALLOWED  
1 NPRCHX) 8 MAXIMUM NUMBER OF PRINT COLUMNS ALLOWED

```

C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      02/02/74      CROP MSAOWH (INTEGER SPACING ONLY)
C      E H SCHLOSSER      LEC      10/26/79      CROP PPDOWH&MSAOWH (SCALING/SPACING)
C
C
C METHOD
C -----
C
C      INITIALIZE TEMPORARY MSA INPUT ENVELOPE.
C      COMPUTE TEMPORARY PPD INPUT ENVELOPE AROUND TEMPORARY MSA INPUT ENVELOPE.
C      INTERSECT PPD OUTPUT ENVELOPE WITH TEMPORARY PPD INPUT ENVELOPE.
C      CROP INTERSECTED PPD OUTPUT ENVELOPE SYMMETRICALLY TO FIT NPRLMX
C      AND NPRCHX.
C      COMPUTE MSA OUTPUT ENVELOPE AROUND CROPPED PPD OUTPUT ENVELOPE.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      MDNOTE      8 PRINT/COUNT/LOG 'NOTE' DIAGNOSTIC MESSAGE
C      MDWARN      8 PRINT/COUNT/LOG 'WARNING' DIAGNOSTIC MESSAGE
C      P4A         8 PRINT/PLOT COORDINATES FOR ADJUSTED MSS COORDINATES
C      A4P         8 ADJUSTED MSS COORDINATES FOR PRINT/PLOT COORDINATES
C
C
C EXCEPTIONS
C -----
C
C      1. IF CALWIN HAS NOT BEEN CALLED PREVIOUSLY, THEN THE RESULTS OF CROPOW
C          ARE UNDEFINED.
C
C      2. IF COMMON MSA INPUT ENVELOPE IS EMPTY, THE TEMPORARY MSA INPUT ENVELOPE
C          IS INITIALIZED TO NOMINAL FULL-SCENE VALUES TO SUPPORT DEBUG USE OF
C          'SYNTHETIC' DATA.
C
C      3. IF THE INTERSECTION OF THE OUTPUT & INPUT ENVELOPES IS EMPTY, A
C          'WARNING' DIAGNOSTIC IS GENERATED.
C
C      4. IF THE PPD OUTPUT ENVELOPE (TRUNCATED TO INTEGER) IS LARGER THAN
C          NPRLMX LINES X NPRCHX COLUMNS, THE ENVELOPE IS CROPPED SYMMETRICALLY
C          AND A 'NOTE' DIAGNOSTIC IS GENERATED.

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

CROPON  
002

```

C
C
C GLOBAL DECLARATIONS
C -----
C
      INCLUDE KOMXQT.LIST      3 COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
      INCLUDE WINDEF.LIST      3 DEFINE STRUCTURE OF WINDOW PACKETS
      INCLUDE KOMIWW.LIST      3 COMMON INPUT WINDOW PACKETS
      INCLUDE KOMOWW.LIST      3 COMMON OUTPUT WINDOW PACKETS
      INCLUDE MAXINT.LIST      3 DEFINE MAXIMUM INTEGER
C
C
C LOCAL DECLARATIONS
C -----
C
      INTEGER MSATIW(2,3)      3 TEMPORARY MSA INPUT WINDOW PACKET
      REAL    PPDITW(2,3)      3 TEMPORARY PPD INPUT WINDOW PACKET
      INTEGER NOD1,NOD2        3 NODE SUBSCRIPTS FOR ENVELOPE
C
C
C PROCEDURE
C -----
C
      CALL TRACE
C
C
C INITIALIZE TEMPORARY MSA INPUT ENVELOPE:
C (FOR FULL-SCENE SYNTHETIC DATA IF COMMON MSA INPUT ENVELOPE IS EMPTY,
C WITH CONTENTS OF COMMON MSA INPUT ENVELOPE OTHERWISE)
C
      MSATIW(WLIN,WMIN)=1
      IF(MSAIWW(WLIN,WMIN).NE.0) MSATIW(WLIN,WMIN)=MSAIWW(WLIN,WMIN)
      MSATIW(WLIN,WMAX)=2400
      IF(MSAIWW(WLIN,WMAX).NE.0) MSATIW(WLIN,WMAX)=MSAIWW(WLIN,WMAX)
      MSATIW(WSAM,WMIN)=1
      IF(MSAIWW(WSAM,WMIN).NE.0) MSATIW(WSAM,WMIN)=MSAIWW(WSAM,WMIN)
      MSATIW(WSAM,WMAX)=3300
      IF(MSAIWW(WSAM,WMAX).NE.0) MSATIW(WSAM,WMAX)=MSAIWW(WSAM,WMAX)
C
C
C COMPUTE TEMPORARY PPD INPUT ENVELOPE AROUND TEMPORARY MSA INPUT ENVELOPE
C
      CALL ENVP4A
C
C
C SET PPD OUTPUT ENVELOPE TO ITS INTERSECTION WITH TEMPORARY PPD INPUT ENVELOPE
C
      PPDOWH(WLIN,WMIN)=AMAX1(PPDOWH(WLIN,WMIN),PPDITW(WLIN,WMIN))
      PPDOWH(WLIN,WMAX)=AMIN1(PPDOWH(WLIN,WMAX),PPDITW(WLIN,WMAX))
      PPDOWH(WCOL,WMIN)=AMAX1(PPDOWH(WCOL,WMIN),PPDITW(WCOL,WMIN))
      PPDOWH(WCOL,WMAX)=AMIN1(PPDOWH(WCOL,WMAX),PPDITW(WCOL,WMAX))
C
C
C CHECK IF INTERSECTED PPD OUTPUT ENVELOPE IS EMPTY
C

```

**CROPON**  
**003**

**N-51**



DAH PACKAGE APPENDIX N  
UTILITY ROUTINES

CROPON  
004

```

330      CONTINUE
360 CONTINUE
C
C
C INSURE PRINT/PLOT COORDINATES ARE ALWAYS POSITIVE
C
      PPDTH(WLIN.WMIN)=AMAX1(PPDTH(WLIN.WMIN),2.)
      PPDTH(WCOL.WMIN)=AMAX1(PPDTH(WCOL.WMIN),2.)
C
C
      RETURN
C
C
C
C
C INTERNAL
SUBROUTINE ENVA4P 3 COMPUTE MSAOWH ENVELOPE AROUND PPDOWH ENVELOPE
C
      REAL ADJLIN,ADJSAM 3 ADJUSTED LINE. SAMPLE OF ENVELOPE CORNERS
C
      MSAOWH(WLIN.WMIN)=+MAXINT
      MSAOWH(WLIN.WMAX)=-MAXINT
      MSAOWH(WSAM.WMIN)=+MAXINT
      MSAOWH(WSAM.WMAX)=-MAXINT
C
      DO 200 NOD1=WMIN,WMAX
        DO 100 NOD2=WMIN,WMAX
          CALL A4P(ADJLIN,ADJSAM,
                  PPDOWH(WLIN,NOD1),PPDOWH(WCOL,NOD2))
          MSAOWH(WLIN.WMIN)=MIN0(MSAOWH(WLIN.WMIN),IFIX(ADJLIN))
          MSAOWH(WLIN.WMAX)=MAX0(MSAOWH(WLIN.WMAX),IFIX(ADJLIN))
          MSAOWH(WSAM.WMIN)=MIN0(MSAOWH(WSAM.WMIN),IFIX(ADJSAM))
          MSAOWH(WSAM.WMAX)=MAX0(MSAOWH(WSAM.WMAX),IFIX(ADJSAM))
100      CONTINUE
200 CONTINUE
C
      RETURN
      END

```

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

DCORLT  
001

SUBROUTINE DCORLT: 8 CORRELATIONS/MEANS/DEVS FROM DB PR SUMS/SUMS-OF-PROD  
O CORREL. 8 SINGLE PRECISION MATRIX OF CORRELATION COEFFICIENTS  
O AMEAN. 8 SINGLE PRECISION VECTOR OF ARITHMETIC MEANS  
O STDEV. 8 SINGLE PRECISION VECTOR OF STANDARD DEVIATIONS  
.  
I NOBS. 8 NUMBER OF OBSERVATIONS (GREATER THAN 500)  
I DSUM. 8 DOUBLE PRECISION VECTOR OF SUMS  
I DSPROD. 8 DOUBLE PRECISION MATRIX OF SUMS OF PRODUCTS  
I NRCUSE. 8 NUMBER OF ROWS & COLUMNS USED  
I NRCDIM 8 NUMBER OF ROWS & COLUMNS DIMENSIONED  
-----

C  
C  
C  
C HISTORY  
C -----  
C  
C E M SCHLOSSER LEC 03/05/78 ORIGINAL CODE  
C E M SCHLOSSER LEC 12/13/79 CHECK FOR BAD DSPROD VALUES  
C  
C

C METHOD  
C -----

C STANDARD STATISTICS.

C MACHINE-DEPENDENT CODE  
C -----

C NONE.

C EXTERNAL REFERENCES  
C -----

C MDNOTE 8 PRINT/COUNT/LOG 'NOTE' DIAGNOSTIC MESSAGE  
C MDWARN 8 PRINT/COUNT/LOG 'WARNING' DIAGNOSTIC MESSAGE  
C DOUBLE PRECISION CBS4IN 8 CHARACTER STRING FOR INTEGER

C EXCEPTIONS  
C -----

C 1. THE FOLLOWING CONDITIONS GENERATE THE DIAGNOSTICS SHOWN:  
C CONDITION DIAGNOSTIC  
C NOBS < 2\*NRCUSE WARNING  
C DSPROD(1,J) < 0 WARNING  
C

C GLOBAL DECLARATIONS  
C -----

C NONE.

C LOCAL DECLARATIONS

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

DCORLT  
002

```

C -----
C
      REAL CORREL(NRCDIM,NRCDIM),AMEAN(NRCDIM),STDEV(NRCDIM)
      DOUBLE PRECISION DSUM(NRCDIM),DSPROD(NRCDIM,NRCDIM)
C
C
C PROCEDURE
C -----
C
      CALL TRACE
C
C
C COMPUTE MEANS & STANDARD DEVIATIONS
C
      IF(NOBS.LT.2*NRCUSE) CALL MDWARN( 'NOBS TOO SMALL IN DCORLT')
      DO 160 NR=1,NRCUSE
        DO 140 NC=1,NRCUSE
          IF(DSPROD(NR,NC).LE.0.) CALL MDWARN(
            'DSPROD('||CB54IN(NR,1)||','||CB54IN(NC,1)||') <= 0 IN DCORLT')
          CORREL(NR,NC)=(DSPROD(NR,NC) - DSUM(NR)*DSUM(NC)/NOBS) / NOBS
140      CONTINUE
160 CONTINUE
        DO 180 NR=1,NRCUSE
          AMEAN(NR)=DSUM(NR)/NOBS
          STDEV(NR)=SQRT(CORREL(NR,NR))
180 CONTINUE
C
C
C COMPUTE CORRELATION COEFFICIENTS
C
      DO 260 NR=1,NRCUSE
        DO 240 NC=1,NRCUSE
          CORREL(NR,NC)=CORREL(NR,NC)/(STDEV(NR)*STDEV(NC))
240      CONTINUE
260 CONTINUE
      RETURN
      END

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

DEG  
001

REAL FUNCTION DEG( 3 DEGREES (FROM DEGREES.MINUTES.SECONDS)  
1 IDEG, 3 WHOLE DEGREES (INTEGER)  
1 IMIN, 3 WHOLE MINUTES (INTEGER)  
1 SEC) 3 SECONDS (REAL)  
-----

C  
C  
C (EHS)  
C  
C

SIG=1DEG  
DEG=FLOAT(1ABS(1DEG))\*FLOAT(1MIN)/60.\*SEC/3600.  
DEG=SIGN(DEG,SIG)  
RETURN  
END

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**DELETE-DENS  
001**

**8DELETE.C \*DANDEN-1.  
8DELETE.C \*DANDEN-2.  
8DELETE.C \*DANDEN-3.  
8DELETE.C \*DANDEN-4.**

```

SUBROUTINE DOJR( 8 DOUBLE PRECISION GAUSS-JORDAN REDUCTION
U A.      8 FIRST N COLUMNS:
C          1: N X N MATRIX      0: N X N INVERSE
C          8 LAST NC-N COLUMNS:
C          1: CONSTANT VECTORS  0: SOLUTION VECTORS
1 NC.      8 NUMBER OF COLUMNS DIMENSIONED IN A
1 NR.      8 NUMBER OF ROWS DIMENSIONED IN A
1 N.       8 NUMBER OF ROWS IN A
1 MC.      8 NUMBER OF COLUMNS IN A
1 S.       8 RETURN TRANSFER LABEL IF OVERFLOW
0 JC.      8 JC(1) = -1*(ROW # OF LAST CORRECTLY REDUCED ROW)
U V1      8 1: OPTIONS REQUESTED: 1. 2. 3. 4. 5. 6. 7.
C          INVERT MATRIX      Y N Y N Y N Y
C          SOLVE DETERMINANT  N Y Y N N Y Y
C          SOLVE SIMUL EQUAT  N N N Y Y Y Y
C          8 0: IF DETERMINANT REQUESTED:
C          V(1) = SIGN OF DETERMINANT
C          V(2) = DLOG(DABS(DETERMINANT))
C          -----
C
DOUBLE PRECISION A(NR,NC)
DOUBLE PRECISION V(2)
INTEGER JC(N)

```

```

C
C
*****
***** UNIVAC 1100 MATHPACK ROUTINE *****
***** SOURCE CODE IS NOT AVAILABLE *****
*****

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

DOPCNT  
001

```

SUBROUTINE DOPCNT( 3 COMPUTE % OF TRACE FOR MATRIX DIAGONAL ELEMENTS
I AMAT,      3 REAL SINGLE PRECISION MATRIX
I NRUSE,     3 NUMBER OF ROWS USED (AMAT)
I NCUSE,     3 NUMBER OF COLUMNS USED (AMAT & PTRAC)
I NRDIM,     3 NUMBER OF ROWS DIMENSIONED (AMAT)
I NCDIM,     3 NUMBER OF COLUMNS DIMENSIONED (AMAT & PTRAC)
O PTRAC)     3 PERCENT OF TRACE:
C              ROW 1 = INDIVIDUAL %
C              ROW 2 = CUMULATIVE %
C -----
C (E H SCHLOSSER)
C
C      REAL AMAT(NRDIM,NCDIM)
C      REAL PTRAC(2,NCDIM)
C      CALL TRACE
C
C      IF(NRUSE.NE.NCUSE) CALL MOWARN( 'MATRIX NOT SQUARE IN DOPCNT')
C      ATRACE=0.
C      DO 200 I=1,NRUSE
C          ATRACE=ATRACE+AMAT(I,I)
200 CONTINUE
C      ATRACE=ATRACE/100.      3 FOR % CALCULATION
C
C      SUM=0.
C      DO 300 I=1,NRUSE
C          PTRAC(I,I)=AMAT(I,I)/ATRACE      3 INDIVIDUAL %
C          SUM=SUM+AMAT(I,I)/ATRACE
C          PTRAC(2,I)=SUM      3 CUMULATIVE %
300 CONTINUE
C      RETURN
C      END

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

DORECP  
001

```
      SUBROUTINE DORECP( 8 TAKE RECIPROCAL OF MATRIX DIAGONAL ELEMENTS
      I AMAT.           8 ORIGINAL SINGLE PRECISION MATRIX
      I NRUSE,NCUSE.    8 NUMBER OF ROWS & COLUMN? USED
      I NRDIM,NCDIM.    8 NUMBER OF ROWS & COLUMNS DIMENSIONED
      O BMAT)          8 NEW MATRIX (MAY BE SAME AS AMAT)
      -----
C
C
C (E H SCHLOSSER)
C
C
C      REAL AMAT(NRDIM,NCDIM),BMAT(NRDIM,NCDIM)
C
C      DO 200 I=1,NRUSE
C          BMAT(I,I)=1./AMAT(I,I)
200 CONTINUE
      RETURN
      END
```



DAH PACKAGE APPENDIX N  
UTILITY ROUTINES

DOSQRT  
001

```
      SUBROUTINE DOSQRT(  & TAKE SQRT OF MATRIX DIAGONAL ELEMENTS  
      I AMAT.             & ORIGINAL SINGLE PRECISION MATRIX  
      I NRUSE,NCUSE.      & NUMBER OF ROWS & COLUMNS USED  
      I NRDIM,NCDIM.      & NUMBER OF ROWS & COLUMNS DIMENSIONED  
      O BMAT)             & NEW MATRIX (MAY BE SAME AS AMAT)  
      -----  
C  
C  
C (E H SCHLOSSER)  
C  
C  
C      REAL AMAT(NRDIM,NCDIM),BMAT(NRDIM,NCDIM)  
C  
      DO 200 I=1,NRUSE  
          BMAT(I,I)=SQRT(AMAT(I,I))  
200 CONTINUE  
      RETURN  
      END
```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

DMPTIC  
001

```

      SUBROUTINE DMPTIC  & DUMP TICK TABLE
      -----
C
C
C (E H SCHLOSSER)
C
C
C      INCLUDE KONTOL.LIST
C
C
C CHECK IF TICK TABLE IS PRESENT
C
      IF(KTBLTY.EQ.'TICK') GO TO 200
      WRITE(6,105)
      105 FORMAT('0 * NO TICK TABLE *')
      GO TO 900
C
C
C DUMP TICK TABLE
C
      200 WRITE(6,205)
      205 FORMAT('0 * * TICK TABLE * *')
      DO 300 NTICK=1,KTBSZ
      IPLTIC=LINTIC(NTICK)
      IPCTIC=COLTIC(NTICK)
      LVL TIC=LEV TIC(NTICK)
      WRITE(6,245) NTICK,IPLTIC,IPCTIC,LVL TIC
      245 FORMAT(14,1X,16,1X,16,1X,12)
      IF(IPLTIC.GT.8000) GO TO 900
      IF((IPLTIC.EQ.0).AND.(IPCTIC.EQ.0).AND.(NTICK.GT.10))
      & GO TO 900
      300 CONTINUE
C
C
      900 RETURN
      END

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

DMPWIN  
001

```

SUBROUTINE DMPWIN( 3 DUMP WINDOW PACKET
1 NAMPKT, 3 NAME OF PACKET (1ST LETTER TELLS IF INTEGER OR REAL)
1 NWHPKT) 3 WINDOW PACKET
-----
C
C
C (E H SCHLOSSER)
C
C
C      DIM      3N NWHPKT(1)
C
C
C
C EXTRACT HEADER FROM WINDOW PACKET
C
      NHEAD1=NWHPKT(1)
      NHEAD2=NWHPKT(2)
      NWDMAX=MINO(NHEAD2*2.50)
      IF(NHEAD1.NE.0) NWDMAX=MINO(NHEAD1*2.NWDMAX)
      IF(NHEAD1.EQ.0) NWDMAX=MINO(14.NWDMAX)
      NWDMAX=MAX0(NWDMAX,6)
C
C
C PRINT NAME AND HEADER OF WINDOW PACKET
C
      WRITE(6,105) NAMPKT
105 FORMAT(
& '0 * * *.A6.' WINDOW PACKET * *)
      WRITE(6,125) NHEAD1,NHEAD2
125 FORMAT(' 01 HEAD ',17,112)
      IF((FLO(0.6,NAMPKT).LT.'000001').OR.
& (FLO(0.6,NAMPKT).GT.'00000N')) GO TO 300      3 REAL WINDOW
C
C
C PRINT BODY OF INTEGER WINDOW PACKET
C
      WRITE(6,245) (NWHPKT(NWD),NWD=3,NWDMAX)
245 FORMAT(
& ' 02 MIN '.17.5X.17/
& ' 03 MAX '.17.5X.17/
& ' 04 INC '.17.5X.17/
& ' 05 TIC+0'.17.5X.17/
& ' 06 TIC+1'.17.5X.17/
& ' 07 ORIO '.17.5X.17/
& ' 08 VER+0'.17.5X.17/
& ' 09 VER+1'.17.5X.17/
& ' 10 VER+2'.17.5X.17/
& ' 11 VER+3'.17.5X.17/
& ' 12 VER+4'.17.5X.17/
& ' 13 VER+5'.17.5X.17/
& ' * VER+*.17.5X.17))
      GO TO 900
C
C
C PRINT BODY OF REAL WINDOW PACKET
C
300 WRITE(6,345) (NWHPKT(NWD),NWD=3,NWDMAX)
345 FORMAT(

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

ONPWIN  
002

```
8 * 02 MIN  *.2F12.4/  
8 * 03 MAX  *.2F12.4/  
8 * 04 INC  *.2F12.4/  
8 * 05 TIC+0*.2F12.4/  
8 * 06 TIC+1*.2F12.4/  
8 * 07 ORIG *.2F12.4/  
8 * 08 VER+0*.2F12.4/  
8 * 09 VER+1*.2F12.4/  
8 * 10 VER+2*.2F12.4/  
8 * 11 VER+3*.2F12.4/  
8 * 12 VER+4*.2F12.4/  
8 * 13 VER+5*.2F12.4/  
8 ( * * VER+* *.2F12.4)
```

C  
C

900 RETURN  
END

**DSSPR**  
**001**

**N-64**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

OSSPR  
002

```

      INTEGER NROW,NCOL      2 ROW/COLUMN NUMBER
C
C
C PROCEDURE
C -----
C
      CALL TRACE
C
C
C INITIALIZE SUMS & SUMS OF PRODUCTS
C
      DO 110 NROW=1,NVARS
          DSUM(NROW)=0.
          DO 100 NCOL=1,NVARS
              DSPROD(NROW,NCOL)=0.
          100 CONTINUE
      110 CONTINUE
C
C
C COMPUTE SUMS & UPPER TRIANGULAR SUMS OF PRODUCTS
C
      DO 140 NOB=1,NOBS
          DO 130 NROW=1,NVARS
              DSUM(NROW)=DSUM(NROW)+TATA(NOB,NROW)
              DO 120 NCOL=NROW,NVARS
                  DSPROD(NROW,NCOL)=DSPROD(NROW,NCOL)+
                      TATA(NOB,NROW)+TATA(NOB,NCOL)
              120 CONTINUE
          130 CONTINUE
      140 CONTINUE
C
C
C COPY UPPER TRIANGULAR SUMS OF PRODUCTS TO LOWER TRIANGULAR
C
      DO 230 NROW=1,NVARS
          DO 220 NCOL=NROW,NVARS
              DSPROD(NCOL,NROW)=DSPROD(NROW,NCOL)
          220 CONTINUE
      230 CONTINUE
C
      RETURN
      END

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

DSSPR3  
001

SUBROUTINE DSSPR3( 3 D P SUMS/SUMS-OF-PRODUCTS FROM MSS TAPE (UNIT 3)  
0 DSUM. 3 DOUBLE PRECISION VECTOR OF SUMS  
0 DSPROD. 3 DOUBLE PRECISION MATRIX OF SUMS OF PRODUCTS  
1 NRCDIM. 3 NUMBER OF ROWS AND COLUMNS IN DSPROD  
0 KPIXLS. 3 NUMBER OF PIXELS  
1 KCHANS) 3 NUMBER OF CHANNELS

C -----

C

C

C HISTORY

C -----

C

C M L BROWN

LEC 01/13/78

ALGORITHM CODING

C J C CRISP

LEC 10/11/79

REVISE SPACING AND PIXEL BUFFER  
AND ADD RESAMPLING AND SCREENING

C

C

C

C METHOD

C -----

C

C INITIALIZE SUMS AND SUMS OF PRODUCTS AND PIXEL COUNTER.

C INITIALIZE LOW AND HIGH LINES AND SAMPLES AND SPACING. POSITION

C AT TOP OF READ WINDOW. READ REQUESTED SCAN LINES AND CHECK FOR

C I/O ERRORS. RESAMPLE AND SCREEN PIXELS FOR EACH LINE. COMPUTE

C SUMS AND UPPER TRIANGULAR SUMS OF PRODUCTS AND COUNT RESAMPLED

C AND SCREENED PIXELS. CONVERT SUMS AND SUMS OF PRODUCTS TO DOUBLE

C PRECISION. COPY UPPER TRIANGULAR SUMS OF PRODUCTS TO LOWER

C TRIANGULAR.

C

C

C MACHINE-DEPENDENT CODE

C -----

C

C NONE

C

C

C EXTERNAL REFERENCES

C -----

C

C GETRAD 3 GET ALL SELECTED RAW/TRANSFORMED CHANNELS

C GETBYT 3 GET BYTE FROM BYTE STRING

C ANP 3 ADJUSTED COORD FOR PRINT/PLOT COORD

C HDFATL 3 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES

C DOUBLE PRECISION C8S4CS 3 VARIABLE-LENGTH CST FOR FIXED LENGTH

C

C

C EXCEPTIONS

C -----

C

C 1. THE FOLLOWING I/O ERRORS GENERATE THE FOLLOWING DIAGNOSTICS:

C 'BADR' FATAL

C 'BADF' FATAL

C 'EOF' FATAL

C 'OFL' FATAL

C

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

DSSPR3  
002

```

C
C GLOBAL DECLARATIONS
C -----
C
      INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES.COUNTERS
      INCLUDE KOMKLS.LIST      & COMMON CLASSIFICATION INFO
      INCLUDE MINDEF.LIST      & DEFINE WINDOW PACKETS
      INCLUDE KOMOWH.LIST      & DEFINE OUTPUT WINDOW PACKET
      INCLUDE PXBDEF.LIST      & DEFINE BUFFER STRUCTURE
      INCLUDE MAXINT.LIST      & MAXIMUM INTEGER VALUE

C
C LOCAL DECLARATIONS
C -----
C
      INTEGERS IN MSA BUF = *INTS PREAMBLE + (*BINS*31)/4 + (*EXTRA BYTES*31)/4
      PARAMETER NMIXBF = (PXBINS-1) + (3548*31)/4 + (19*31)/4
      PARAMETER NXBUFS=8      & NUMBER OF MSS PIXEL BUFFERS IN ARRAY
      DOUBLE PRECISION DSUM(1)      & ARGUMENT
      DOUBLE PRECISION DSPROD(NRCDIM,NRCDIM)      & ARGUMENT
      INTEGER MPXBUF(NMIXBF,NXBUFS)      & ARRAY OF PIXEL BUFFERS
      INTEGER ISUM(NXBUFS)      & TEMPORARY ARRAY OF SUMS
      INTEGER ISPROD(NXBUFS,NXBUFS)      & TEMPORARY MATRIX OF SUMS OF PRODUCTS
      INTEGER IPIXEL(NXBUFS)      & PIXEL VALUE FOR EACH BUFFER
      INTEGER NBINS0(NXBUFS)      & BIN # OF SAMPLE 0 FOR EACH BUFFER
      INTEGER ISTAT      & I/O STATUS
      INTEGER NROW,NCOL      & ROW/COLUMN NUMBERS
      INTEGER MSALIN,MSASAM      & MSA LINE AND SAMPLE
      INTEGER MSASLO,MSASHI      & MSA LOW AND HIGH SAMPLE
      REAL ADJLIN,ADJSAM      & ADJUSTED LINE AND SAMPLE
      INTEGER ML100L,ML100H,ML100S      & MSA LINE*100: LOW,HIGH,SPACING
      INTEGER MS100L,MS100H,MS100S      & MSA SAMPLE*100: LOW,HIGH,SPACING
      INTEGER LASTLN      & LAST LINE READ
      INTEGER IPLIN      & PRINT LINE
      INTEGER IPCHIN,IPCHAX      & MINIMUM AND MAXIMUM PRINT COLUMN
      INTEGER IPLMIN,IPLMAX      & MINIMUM AND MAXIMUM PRINT LINE

C
C PROCEDURE
C -----
C
      CALL TRACE

C
C INITIALIZE MINIMUM AND MAXIMUM PRINT LINES AND COLUMNS
C
      IPLMIN=PPDOWN(WLIN,WMIN)
      IPLMAX=PPDOWN(WLIN,WMAX)
      IPCHIN=PPDOWN(WCOL,WMIN)
      IPCHAX=PPDOWN(WCOL,WMAX)

C
C INITIALIZE SUMS AND SUMS OF PRODUCTS AND PIXEL COUNTER
C
      DO 110 NROW=1,KCHANS
        DSUM(NROW)=0.

```



DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

DSSPR3  
003

```

        DO 100 NCOL=1,KCHANS
            DSPROD(NROW,NCOL)=0.
100      CONTINUE
110      CONTINUE
        KPIXLS=0
C
C
C INITIALIZE LOW AND HIGH LINES AND SPACING
C
        CALL A4P (ADJLIN,ADJSAM,    FLOAT(IPLMIN),1.)
        ML100L=ADJLIN*100.
        CALL A4P (ADJLIN,ADJSAM,    FLOAT(IPLMAX),1.)
        ML100H=ADJLIN*100.
        ML100S=MSAOWH(MLIN,WSP100)
C
C
C POSITION AT TOP OF WINDOW
C
        MSALIN=ML100L/100
        CALL GETRAD (MPXBUF,(0),(NXBUFS),ISTAT,    MSALIN,0.0)
        IF (ISTAT.NE.'BADF') GO TO 200
        CALL MDFATL ('BADF (BAD FILE) ON UNIT 3')
        GO TO 900
C
C
C READ REQUESTED SCAN LINES
C
200      IPLIN=IPLMIN
        LASTLN=-MAXINT
        DO 730 ML100=ML100L,ML100H,ML100S
            DO 240 NROW=1,KCHANS      & CLEAR TEMP SUMS AND SUMS OF PRODS
                ISUM(NROW)=0
                DO 220 NCOL=1,KCHANS
                    ISPROD(NROW,NCOL)=0
220          CONTINUE
240          CONTINUE
            MSALIN=ML100/100
            CALL A4P (ADJLIN,ADJSAM,    FLOAT(IPLIN),FLOAT(IPCMIN))
            MS100L=ADJSAM*100.
            CALL A4P (ADJLIN,ADJSAM,    FLOAT(IPLIN),FLOAT(IPCMAX))
            MS100H=ADJSAM*100.
            MS100S=MSAOWH(MSAM,WSP100)
            MSASLO=MS100L/100
            MSASHI=MS100H/100
            IF (MSALIN.NE.LASTLN) CALL GETRAD (MPXBUF,(NWXBF),(NXBUFS),
                & ISTAT,    MSALIN,MSASLO,MSASHI)
            LASTLN=MSALIN
            IF (ISTAT.EQ.' ') GO TO 260
            CALL MDFATL (CBS4CS(ISTAT,1,4),
                & ' WHILE READING ON UNIT 3')
            GO TO 900
260          CALL MSKPIX (MPXBUF(1,1),    MPXBUF(1,1))
C
C
C INITIALIZE BIN NUMBER FOR SAMPLE 0 FOR EACH BUFFER
C

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

DSSPR3  
004

```

      DO 290 NUMBUF=1,KCHANS
        NBINSO(NUMBUF)=MPXBUF(PXLBIN,NUMBUF)
        -MPXBUF(PXLSAM,NUMBUF)
      290 CONTINUE
C
C
C RESAMPLE.SCREEN.COMPUTE SUMS AND SUMS OF PRODS FOR EACH LINE
C
      DO 650 MS100=MS100L,MS100H,MS170S
        MSASAM=MS100/100
C
C BUFFER 1
C
      IF ((MSASAM.LT.MPXBUF(PXLSAM,1)).OR.
        (MSASAM.GT.MPXBUF(PXHSAH,1))) GO TO 650      3 SAM NOT IN BUF
      CALL GETBYT (IPIXEL(1),
        MPXBUF(PXBINS,1),(MSASAM+NBINSO(1)))
      IF ((IPIXEL(1).GE.MPXBUF(PXNODA,1)) GO TO 650      3 NO DATA
      IF ((IPIXEL(1).LT.LCVLO(1)).OR.
        (IPIXEL(1).GT.LCVHI(1))) GO TO 650      3 OUT OF RAD LIM
      IF (KCHANS-1.EQ.0) GO TO 590
C
C BUFFER 2
C
      IF ((MSASAM.LT.MPXBUF(PXLSAM,2)).OR.
        (MSASAM.GT.MPXBUF(PXHSAH,2))) GO TO 650      3 SAM NOT IN BUF
      CALL GETBYT (IPIXEL(2),
        MPXBUF(PXBINS,2),(MSASAM+NBINSO(2)))
      IF ((IPIXEL(2).LT.LCVLO(2)).OR.
        (IPIXEL(2).GT.LCVHI(2))) GO TO 650      3 OUT OF RAD LIM
      IF (KCHANS-2.EQ.0) GO TO 530
C
C BUFFER 3
C
      IF ((MSASAM.LT.MPXBUF(PXLSAM,3)).OR.
        (MSASAM.GT.MPXBUF(PXHSAH,3))) GO TO 650      3 SAM NOT IN BUF
      CALL GETBYT (IPIXEL(3),
        MPXBUF(PXBINS,3),(MSASAM+NBINSO(3)))
      IF ((IPIXEL(3).LT.LCVLO(3)).OR.
        (IPIXEL(3).GT.LCVHI(3))) GO TO 650      3 OUT OF RAD LIM
      IF (KCHANS-3.EQ.0) GO TO 470
C
C BUFFER 4
C
      IF ((MSASAM.LT.MPXBUF(PXLSAM,4)).OR.
        (MSASAM.GT.MPXBUF(PXHSAH,4))) GO TO 650      3 SAM NOT IN BUF
      CALL GETBYT (IPIXEL(4),
        MPXBUF(PXBINS,4),(MSASAM+NBINSO(4)))
      IF ((IPIXEL(4).LT.LCVLO(4)).OR.
        (IPIXEL(4).GT.LCVHI(4))) GO TO 650      3 OUT OF RAD LIM
      IF (KCHANS-4.EQ.0) GO TO 410
C
C BUFFER 5
C
      IF ((MSASAM.LT.MPXBUF(PXLSAM,5)).OR.
        (MSASAM.GT.MPXBUF(PXHSAH,5))) GO TO 650      3 SAM NOT IN BUF

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

DSSPR3  
005

```

      CALL GETBYT (IPIXEL(5),
      *      MPXBUF(PXBINS,5),(MSASAM+NBINSO(5)))
      IF ((IPIXEL(5).LT.LCVLO(5)).OR.
      &      (IPIXEL(5).GT.LCVHI(5))) GO TO 650      & OUT OF RAD LIM
      IF (KCHANS-S.EQ.0) GO TO 350

C
C BUFFER 6
C
      IF ((MSASAM.LT.MPXBUF(PXLSAM,6)).OR.
      &      (MSASAM.GT.MPXBUF(PXHSAM,6))) GO TO 650      & SAM NOT IN BUF
      CALL GETBYT (IPIXEL(6),
      *      MPXBUF(PXBINS,6),(MSASAM+NBINSO(6)))
      IF ((IPIXEL(6).LT.LCVLO(6)).OR.
      &      (IPIXEL(6).GT.LCVHI(6))) GO TO 650      & OUT OF RAD LIM

C
C COMPUTE SUMS AND UPPER TRIANGULAR SUMS OF PRODUCTS
C
      ISUM(6)=ISUM(6)+IPIXEL(6)
      DO 320 NCOL=6,KCHANS
      ISPROD(6,NCOL)=ISPROD(6,NCOL)+IPIXEL(6)*IPIXEL(NCOL)
320      CONTINUE

C
C
350      ISUM(5)=ISUM(5)+IPIXEL(5)
      DO 380 NCOL=5,KCHANS
      ISPROD(5,NCOL)=ISPROD(5,NCOL)+IPIXEL(5)*IPIXEL(NCOL)
380      CONTINUE

C
C
410      ISUM(4)=ISUM(4)+IPIXEL(4)
      DO 440 NCOL=4,KCHANS
      ISPROD(4,NCOL)=ISPROD(4,NCOL)+IPIXEL(4)*IPIXEL(NCOL)
440      CONTINUE

C
C
470      ISUM(3)=ISUM(3)+IPIXEL(3)
      DO 500 NCOL=3,KCHANS
      ISPROD(3,NCOL)=ISPROD(3,NCOL)+IPIXEL(3)*IPIXEL(NCOL)
500      CONTINUE

C
C
530      ISUM(2)=ISUM(2)+IPIXEL(2)
      DO 560 NCOL=2,KCHANS
      ISPROD(2,NCOL)=ISPROD(2,NCOL)+IPIXEL(2)*IPIXEL(NCOL)
560      CONTINUE

C
C
590      ISUM(1)=ISUM(1)+IPIXEL(1)
      DO 620 NCOL=1,KCHANS
      ISPROD(1,NCOL)=ISPROD(1,NCOL)+IPIXEL(1)*IPIXEL(NCOL)
620      CONTINUE
      KPIXLS=KPIXLS+1
650      CONTINUE

C
C
C CONVERT TO DOUBLE PRECISION

```

**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**DSSPR3  
006**

```

C
      DO 700 NROW=1,KCHANS
        DSUM(NROW)=DSUM(NROW)+ISUM(NROW)
        DO 680 NCOL=NROW,KCHANS
          DSPROD(NROW,NCOL)=DSPROD(NROW,NCOL)+
            ISPROD(NROW,NCOL)
        680
      680      CONTINUE
      700      CONTINUE
            IPLIN=IPLIN+1
      730 CONTINUE
C
C
C COPY UPPER TRIANGULAR SUMS OF PRODUCTS TO LOWER TRIANGULAR
C
      DO 750 NROW=1,KCHANS
        DO 750 NCOL=NROW,KCHANS
          DSPROD(NCOL,NROW)=DSPROD(NROW,NCOL)
      750      CONTINUE
      790 CONTINUE
C
C
C
      900 RETURN
      END

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

02DMS  
001

```
      SUBROUTINE 02DMS( 3 DEGREES TO DEGREES.MINUTES.SECONDS
1 DEO.      3 DEGREES (REAL)
0 IDEO.     3 WHOLE DEGREES (INTEGER)
0 IMIN.     3 WHOLE MINUTES (INTEGER)
0 SEC)      3 SECONDS (REAL)
-----
C
C
C (CMS)
C
      IDEO=IFIX(DEO)
      AMIN=ABS(DEO-FLOAT(IDEO))*60.
      IMIN=AMIN
      SEC=(AMIN-FLOAT(IMIN))*60.
      RETURN
      END
```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

EISRTD  
001

SUBROUTINE EISRTD: 8 SORT E-VALS/E-VECS IN DESCENDING ORDER BY E-VAL  
U EIOVAL. 8 MATRIX WITH EIGENVALUES 44 DIAGONAL  
U EIOVEC. 8 MATRIX OF EIGENVECTORS  
I NRCUSE. 8 NUMBER OF ROWS 8 COLUMNS USED  
I NRCDIM) 8 NUMBER OF ROWS 8 COLUMNS DIMENSIONED

```

C
C
C (E H SCHLOSSER)
C
C
C THIS SUBROUTINE SORTS IN DESCENDING SEQUENCE USING SHUTTLESORT
C (CACH ALGORITHM # 179). THIS TECHNIQUE IS VERY EFFICIENT WHEN (AND ONLY WHEN)
C THE DATA IS ALREADY GROSSLY SORTED IN THE PROPER SEQUENCE.
C
      REAL EIOVAL(NRCDIM,NRCDIM),EIOVEC(NRCDIM,NRCDIM)
      CALL TRACE
C
C
      IF(NRCUSE.LT.2) GO TO 900
      NI=NRCUSE-1
      DO 400 I=1,NI
        DO 300 NC=1,I.-1
          IF(EIOVAL(NC,NC).GE.EIOVAL(NC+1,NC+1)) GOTO 400
          RLTEMP=EIOVAL(NC,NC)
          EIOVAL(NC,NC)=EIOVAL(NC+1,NC+1)
          EIOVAL(NC+1,NC+1)=RLTEMP
          DO 200 NR=1,NRCUSE
            RLTEMP=EIOVEC(NR,NC)
            EIOVEC(NR,NC)=EIOVEC(NR,NC+1)
            EIOVEC(NR,NC+1)=RLTEMP
          CONTINUE
        CONTINUE
      CONTINUE
      200
      300
      400
      900 RETURN
      END

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

ENVORI  
001

SUBROUTINE ENVORI( 8 ADD ORIGIN TO ENVELOPE (REAL ONLY)  
U AWWPKT) 8 WINDOW PACKET (REAL)

C  
C  
C  
C  
C

REAL AWWPKT(2,1)  
INCLUDE WINDEF.LIST

C  
C

AWWPKT(1,WMIN)=AWWPKT(1,WMIN)+AWWPKT(1,WORIG)  
AWWPKT(1,WMAX)=AWWPKT(1,WMAX)+AWWPKT(1,WORIG)  
AWWPKT(2,WMIN)=AWWPKT(2,WMIN)+AWWPKT(2,WORIG)  
AWWPKT(2,WMAX)=AWWPKT(2,WMAX)+AWWPKT(2,WORIG)  
RETURN  
END

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

ENVWIN  
001

```

      SUBROUTINE ENVWIN( 8 COMPUTE ENVELOPE FOR REAL WINDOW
      U EHPKPT) 8 WINDOW PACKET (REAL)
      -----
C
C
C (E H. SCHLOSSER)
C
C
C EXTERNAL SUBROUTINES/FUNCTIONS CALLED
C -----
C
C      MOFATL
C
C
C      REAL EHPKPT(2,1)
C      INCLUDE MINDEF.LIST
C      EQUIVALENCE (AXVER,MAXVER)
C
C
C INITIALIZE ENVELOPE
C
C      EHPKPT(1,WMIN)=+9E+35
C      EHPKPT(1,WMAX)=-9E+35
C      EHPKPT(2,WMIN)=+9E+35
C      EHPKPT(2,WMAX)=-9E+35
C
C
C GET POINTERS TO FIRST AND LAST VERTICES
C
C      MINVER=MVER+1
C      AXVER=EHPKPT(MUSED,WHEAD)
C      IF(MAXVER.GT.MINVER) GO TO 200
C      IF(MAXVER.NE.MINVER) CALL MOFATL(
C      *      'BAD NODE POINTER IN ENVWIN')
C      MAXVER=MINVER+1
C      EHPKPT(1,MAXVER)=0.      8 THE OTHER VERTEX IS THE ORIGIN
C      EHPKPT(2,MAXVER)=0.
C
C 200 CONTINUE
C
C
C FIND THE ENVELOPE
C
C      DO 300 NVER=MINVER,MAXVER
C          EHPKPT(1,WMIN)=AMIN1(EHPKPT(1,WMIN),EHPKPT(1,NVER))
C          EHPKPT(1,WMAX)=AMAX1(EHPKPT(1,WMAX),EHPKPT(1,NVER))
C          EHPKPT(2,WMIN)=AMIN1(EHPKPT(2,WMIN),EHPKPT(2,NVER))
C          EHPKPT(2,WMAX)=AMAX1(EHPKPT(2,WMAX),EHPKPT(2,NVER))
C
C 300 CONTINUE
C      RETURN
C      END

```



**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**EOF  
001**

**8EOF . (DAM.EOF)**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

FACPRT  
001

```

SUBROUTINE FACPRT( 3 PRINT FACTOR STRUCTURE/COEFFICIENTS/MEANS
1 FSTRUC.          3 FACTOR STRUCTURE
1 FCNORM.          3 FACTOR COEFFICIENTS OF NORMALIZED CHANNELS
0 FCORIG.          3 FACTOR COEFFICIENTS OF ORIGINAL CHANNELS
1 STD.            3 VECTOR OF STANDARD DEVIATIONS OF ORIG CHANNELS
1 AMEAN.          3 VECTOR OF MEANS OF ORIGINAL CHANNELS
1 NRUSE.NCUSE.     3 NUMBER OF ROWS & COLUMNS USED
1 NRDIM.NCDIM)    3 NUMBER OF ROWS & COLUMNS DIMENSIONED
-----
C
C
C (E H SCHLOSSER)
C
C
C      REAL FSTRUC(NRDIM,NCDIM),FCNORM(NRDIM,NCDIM)
C      REAL FCORIG(NRDIM,NCDIM),FMEAN(7)
C      REAL STD(NCDIM),AMEAN(NCDIM)
C      CALL TRACE
C
C
C PRINT FACTOR STRUCTURE & COEFFICIENTS
C
C      WRITE(6,225)
C 225 FORMAT(' FACTOR STRUCTURE --',
C      & ' CORRELATION (LOADING) BETWEEN CHANNELS & FACTORS')
C      CALL MATPRT(6,FSTRUC,' CHAN',NRUSE,NCUSE,NRDIM,NCDIM)
C      WRITE(6,325)
C 325 FORMAT
C      & (' NORMALIZED FACTORS -- COEFFICIENTS OF NORMALIZED CHANNELS')
C      CALL MATPRT(6,FCNORM,' CHAN',NRUSE,NCUSE,NRDIM,NCDIM)
C
C
C UN-NORMALIZE AND PRINT FACTOR COEFFICIENTS & FACTOR MEANS
C
C      DO 520 NC=1,NRUSE
C          FMEAN(NC)=0.
C          DO 500 NR=1,NCUSE
C              FCORIG(NR,NC)=FCNORM(NR,NC)/STD(NR)
C              FMEAN(NC)=FMEAN(NC)+(FCNORM(NR,NC)/STD(NR))*AMEAN(NR)
C 500      CONTINUE
C 520 CONTINUE
C      WRITE(6,525)
C 525 FORMAT(' BIASED FACTORS -- COEFFICIENTS OF ORIGINAL CHANNELS')
C      CALL MATPRT(6,FCORIG,' CHAN',NRUSE,NCUSE,NRDIM,NCDIM)
C      WRITE(6,545)
C 545 FORMAT(' BIASED FACTORS -- MEANS (STD DEV = 1.0)')
C      CALL MATPRT(6,FMEAN,' '.1,NCUSE,1,NCDIM)
C
C
C      RETURN
C      END

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

FLINFO  
001

SUBROUTINE FLINFO( 8 GET FILE INFORMATION  
0 IDFILE. 8 INFORMATION BUFFER IN FIDEF FORMAT

1 NAMFIL. 8 INTERNAL FILENAME: STRING 1 THRU 12 CHARACTERS FROM  
THE SET A THRU Z, 0 THRU 9 AND DASH (-). IF A VARIABLE  
INSTEAD OF A QUOTED LITERAL, MUST BE PADDED WITH BLANKS  
TO A TOTAL LENGTH OF 12 CHARACTERS.

1 IEXFMT) 8 EXTERNAL FORMAT IF TAPE: 'BB'/'BST'/'CST'

HISTORY

MARY TOMPKINS	LEC	08/08/79	REQUIREMENTS
C A HELMKE	LEC	09/20/79	ALGORITHM DESIGN
C A HELMKE	LEC	09/25/79	ALGORITHM CODING
MARY TOMPKINS	LEMSCO	09/23/80	CALL .. ERINFO

METHOD

INITIALIZE IDFILE. ESTABLISH 'NAMFIL' PACKET TO OBTAIN  
INFORMATION ON FILE OR FACILITY ASSIGNMENT. DEPENDENT UPON  
INPUT DEVICE USED. GENERATE IDFILE AS DEFINED BY THE FIDEF  
FORMAT.  
IF TAPE, SET IDFILE(FIDBFH):

		IEXFMT (EXTERNAL TAPE FORMAT)			
		'BB'	'BST'	'CST'	ANYTHING ELSE
UNIVAC DATA	QWD	'BST'	'ERR'	'ERR'	'ERR'
TRANSFER	BB	'CST'	'ERR'	'ERR'	'ERR'
FORMAT	BB	'BB'	'BST'	'CST'	'ERR'

MACHINE-DEPENDENT CODE

ALIGNMENT OF SCALED INTEGERS ASSUMES 36-BIT COMPUTER WORD.

EXTERNAL REFERENCES

ERFITH 8 RETRIEVE FACILITIES ASSIGNMENT INFORMATION  
ERINFO 8 RETRIEVE SYSTEM.RUN.PROGRAM. & FILE PARAMETERS  
MOVINT 8 MOVE INTEGER FROM ONE STRING TO ANOTHER STRING  
MOVCST 8 MOVE CHARACTER STRING  
INTEGER LENCST 8 LENGTH OF CHARACTER STRING  
DOUBLE PRECISION CDS4IN 8 VARIABLE LENGTH CHAR STRING FOR INTEGER

EXCEPTIONS

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

FLINFO  
002

```

C
C      1.THE FOLLOWING CONDITIONS GENERATE THE EQUIVALENT CODE SHOWN:
C      CONDITIONS                                IDFILE(FIDEQT)
C      FILE NOT ASSIGNED                        'NUL'
C      FILE ASSIGNED TO UNSUPPORTED DEVICE     'OTHR'
C      FILE NAME SYNTAX ERROR                  'ERR'
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES. COUNTERS
C      INCLUDE FIDEF.LIST      & MNEMONICS FOR POINTERS TO LOCATIONS IN
C                                & IDFILE BUFFER.
C      INCLUDE ASMDEF.LIST     & MAKES UNIVAC 1100 ASSEMBLER PARTIAL WORD
C                                & MNEMONICS AVAILABLE TO FORTRAN V PROGRAMS
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER IDFILE(10)      & ARGUMENT
C      INTEGER NAMFIL(2)       & ARGUMENT
C      INTEGER IPAKET(13)      & PACKET FOR ERFITH
C      INTEGER IBLANK/' ' /
C      INTEGER J79EQP(15) /    & ARRAY CONTAINING TAPE TRACK AND EQUIP
C                                & CODE MNEMONICS FOR EQUIP CODES 1-15
C
C      1 '7 8C ' .
C      2 '7 8C ' .
C      3 '7 8CB' .
C      4 '7 8CB' .
C      5 '9 8C9' .
C      6 '9 8C9' .
C      7 '7 4C ' .
C      8 '7 4CB' .
C      9 '7 12 ' .
C      A '7 16 ' .
C      B '9 12N' .
C      C '9 16N' .
C      D '9 20N' .
C      E '7 3A ' .
C      F '7 2A ' /
C
C      INTEGER JDENS(6) / '200'.'556' . & TAPE DENSITIES
C      1 '800'.'1600'.'3200'.'???' /
C      INTEGER IDTFCO          & UNIVAC DATA TRANSFER CODE (IBUFMT ROW INDEX)
C                                1 = QUARTER WORD
C                                2 = 8-BIT PACKED
C                                3 = 8-BIT PACKED
C
C      INTEGER IEXFCO          & EXTERNAL FORMAT CODE 1=BB. 2=BST. 3=CST.
C                                & 4=ANYTHING ELSE (IBUFMT COLUMN INDEX)
C      INTEGER IBUFMT(3,4) /   & LOOKUP TABLE FOR BUFFER FORMAT DEPENDING
C      1 'BST'.'CST'.'BB' .    & UPON EXTERNAL FORMAT AND UNIVAC
C      2 'ERR'.'ERR'.'BST' .   & MSA DATA TRANSFER FORMAT
C      3 'ERR'.'ERR'.'CST' .
C      4 'ERR'.'ERR'.'ERR' /
C      INTEGER NOENS           & INDEX TO JDENS ARRAY

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

FLINFO  
003

```

      INTEGER NEQUIP      & EQUIPMENT CODE
      INTEGER NSUBCD      & EQUIPMENT SUBCODE
      INTEGER INDENS(5)/  & TAPE DENSITIES FROM ERINFO
1  '200','550','800','1600','6250'/
      INTEGER IPKT        & PACKET FOR FUNCTION MODESS IN ERINFO
      INTEGER ISTFIL      & STATUS OF INTERNAL FILE
      INTEGER ISTFUN      & STATUS OF FUNCTION
      INTEGER KNTFUN      & COUNT OF WORSE REQ. TO TRANSMIT ALL DATA

```

```

C
C
C PROCEDURE
C -----
C
C      CALL TRACE
C
C
C      INITIALIZE INFORMATION BUFFER
C
C      DO 100 I=1,7
100  IDFILE(I)=IBLANK
      IDFILE(8)=0
C
C
C      SET UP A PACKET FOR CALL TO ERFITH (ERFITH IS FORTRAN DRIVER
C      WHICH CALLS UNIVAC EXECUTIVE REQUEST ITEMS)
C
C      CALL MOVCST(IPAKET,(1),(12),
C      =          NAMFIL,(1),(LENCST(NAMFIL,12)), ' ')
C      CALL ERFITH(IPAKET)
C
C
C      CHECK FOR VALID FILENAME
C
C      NEQUIP=ASHS1(IPAKET(7))
C      IF (NEQUIP.EQ.0 .AND. IPAKET(1).EQ.IBLANK .AND.
1  IPAKET(2).EQ.IBLANK) GO TO 400 & FILE NAME SYNTAX ERROR
C      IF (NEQUIP.GT.0) GO TO 150 & FILE IS ASSIGNED
C      IDFILE(FIDEQT)='NUL' & FILE NOT ASSIGNED
C      GO TO 900
C
C
C
C      CHECK IF EQUIPMENT TYPE IS TAPE OR DISK
C
C      150 IF (NEQUIP.GE.1 .AND. NEQUIP.LE.15) GO TO 160 & TAPE
C      IF (NEQUIP.GE.24 .AND. NEQUIP.LE.31 .AND. NEQUIP.NE.25)
1  GO TO 300 & FASTRAND DRUM
C      IDFILE(FIDEQT)='OTHR' & UNSUPPORTED DEVICE
C      GO TO 900
C
C
C
C      EQUIPMENT TYPE IS TAPE
C
C      160 IDFILE(FIDEQT)='TAPE'
C      CALL MOVCST (IDFILE(FIDEQT),1,8,J79EQP(NEQUIP),4,3,IBLANK)
C      CALL MOVCST (IDFILE(FIDTRK),1,8,J79EQP(NEQUIP),1,3,IBLANK)
C

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

FLINFO  
004

```

C
C EXTRACT DENSITY AND MSA DATA TRANSFER FORMAT FROM FITEMS PAKET
C
    IF (NEQUIP.07.0) 00 TO 100
    NDENS = FLD(35-13.2,1PAKET(9))      8 FOR TAPE UNITS OTHER THAN
    IDTFCD=3                             8 UNISERVO 12/14/16/20
    00 TO 230
  100 IF (NEQUIP.07.10) 00 TO 200
    NDENS = FLD(35-15.2,1PAKET(9)) + 1  8 FOR 7-TRACK UNISERVO 12/14/16
    00 TO 210
  200 IF (NEQUIP.07.13) 00 TO 220
    NDENS = IABS (FLD(35-11.1,1PAKET(9))-4)  8 FOR 9-TRACK UNISERVO
  210 IDTFCC = FLD (35-17.2,1PAKET(9))+1    8 12/14/16/20/30/32/34/36
    IF (IDTFCD.07.3) IDTFCD=3
    00 TO 230
  220 NDENS=6                               8 FOR UNISERVO 111A AND 11A
    IDTFCD=3
  230 IDFILE (FIDFPI) = JDENS(NDENS)
C
C
C USING EXTERNAL FORMAT AND MSA DATA TRANSFER FORMAT
C LOOK UP BUFFER FORMAT
C
    IF (IEXFMT.NE.'BB') 00 TO 240
    IEXFCD=1
    00 TO 270
  240 IF (IEXFMT.NE.'BST') 00 TO 250
    IEXFCD=2
    00 TO 270
  250 IF (IEXFMT.NE.'CST') 00 TO 260
    IEXFCD=3
    00 TO 270
  260 IEXFCD=4
  270 IDFILE (FIDBFH) = IBUFMT(IDTFCD,IEXFCD)
C
C EXTRACT CURRENT REEL NUMBER AND NEXT REEL NUMBER FROM FITEMS PAKET
C
    IDFILE(FIDCRL)=1PAKET(12)
    IDFILE(FIDNRL)=1PAKET(13)
    IF (IDFILE(FIDNRL).EQ.'000000') IDFILE(FIDNRL)=' '
C
C
C EXTRACT AND CHECK NOISE CONSTANT. INSURE THAT TAPE BLOCKS LESS
C THAN 8 CHARACTERS ARE TREATED AS 'NOISE'
C
    IDFILE(FIDNOS)=ASMS3(1PAKET(9))
    IF (IDFILE(FIDNOS).GE.8) 00 TO 280
    IDFILE (FIDNOS)=8
    CALL ERCSF (NAD,'MODE 3..8 . ')
C
C
C CALL ERINFO. IF STATUS RETURNED IS ACCEPTABLE UPDATE IDFILE.
C IF STATUS REFLECTS A PROGRAMMER ERROR ISSUE DIAGNOSTIC.
C
  280 CALL ERINFO(1STFIL,1STFUN,KNTFUN,1PKT.1, 1PAKET,10)

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

FLINFO  
005

```

      IF(1STFIL.NE.0.AND.1STFIL.NE.2.AND.1STFIL.NE.3) CALL MDWARN(
& 'PROGRAMMER ERROR: FILE STATUS '.COS4IN(1STFIL.2).
& ' IN ERINFO')
      IF((1STFIL.EQ.0.OR.1STFIL.EQ.6).AND.
& (1STFUN.EQ.0.OR.1STFUN.EQ.6.OR.1STFUN.EQ.7)) GO TO 290
      IF(1STFUN.NE.2.AND.1STFUN.NE.3) CALL MDWARN(
& 'PROGRAMMER ERROR: FUNCTION STATUS '.COS4IN(1STFUN.2).
& ' IN ERINFO')
      GO TO 900
290  IDTFCD = FLD(35-3.3,1PKT) & 8/8/QUARTER PACKED
      IDTFCD = MIN0(IDTFCD,3)
      IDFILE(FIDCFH) = I8UFHT(IDTFCD,1EXFCD)
      NDENS = ASH55(1PKT)
      IF(IDFILE(FIDTRK).EQ.'9') NDENS = NDENS +2
      IDFILE(FIDFPI) = INDENS(NDENS) & DENSITY
      GO TO 900
C
C
C EQUIPMENT TYPE IS DISK (FASTRAND DRUM).  USE EQUIPMENT CODE
C AND SUBCODE TO DETERMINE EQUIPMENT CODE MNEMONIC.
C
300  IDFILE(FIDEQT)='DISK'
      IEQCAS=NEQUIP-23
      GO TO (310,320,330,340,350,360,370,380),IEQCAS
310  NSUBCD=ASH56(IPAKET(11)) & EQUIP CODE 24
      NSBCAS=NSUBCD+1
      NSBCAS=MIN0(3,MAX0(1,NSBCAS))
      GO TO (312,312,316),NSBCAS
312  IDFILE(FIDEQC)='F2' & SUBCODE 00 OR 01
      GO TO 900
316  IDFILE(FIDEQC)='F60' & SUBCODE 02
      GO TO 900
320  GO TO 900 & EQUIP CODE 25 NOT USED
330  IDFILE(FIDEQC)='F4' & EQUIP CODE 26
      GO TO 900
340  IDFILE(FIDEQC)='F8' & EQUIP CODE 27
      GO TO 900
350  IDFILE(FIDEQC)='F17' & EQUIP CODE 28
      GO TO 900
360  NSUBCD=ASH56(IPAKET(11)) & EQUIP CODE 29
      NSBCAS=NSUBCD+1
      NSBCAS=MIN0(3,MAX0(1,NSBCAS))
      GO TO (362,364,366),NSBCAS
362  IDFILE(FIDEQC)='F14' & SUBCODE 00
      GO TO 900
364  IDFILE(FIDEQC)='F24' & SUBCODE 01
      GO TO 900
366  IDFILE(FIDEQC)='F25' & SUBCODE 02
      GO TO 900
370  IDFILE(FIDEQC)='F40' & EQUIP CODE 30
      GO TO 900
380  IDFILE(FIDEQC)='FCS' & EQUIP CODE 31
      GO TO 900
C
C
C FILE NAME SYNTAX ERROR

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**FLINFO  
000**

**C  
400 IDFILE(FIDCQT)='ERR'  
C  
C  
C TERMINATE  
C  
900 RETURN  
END**



DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

OCERT  
001

SUBROUTINE OCERT    & ASSIGN/COMPUTE CONSTANTS FOR ERTS GEOMETRY

```

C -----
C
C HISTORY
C -----
C      E M SCHLOSSER      LEC      08/17/73      ORIGINAL CODE
C      MARY TOMPKINS      LEC      10/31/79      TEMP LANSAT-3 CONSTANTS
C
C METHOD
C -----
C      ACCORDING TO NERTS(1) ASSIGN/COMPUTE APPROPRIATE CONSTANTS.
C
C EXTERNAL REFERENCES
C -----
C      MCFATL        & PRINT/COUNT/LOG 'FATAL ERROR' DIAGNOSTIC MESSAGE.
C
C EXCEPTIONS
C -----
C      1. IF NERTS(1) IS INVALID THEN GENERATE FATAL DIAGNOSTIC AND USE
C         LANDSAT-1 CONSTANTS.
C
C GLOBAL DECLARATIONS
C -----
C      INCLUDE KOMNER.LIST        & COMMON ERTS SCENE PARAMETERS
C      INCLUDE KOMF.T.LIST       & COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C
C LOCAL DECLARATIONS
C -----
C      REAL RADEG/.0174532921,        & RADIANS PER DEGREE
C      EQUIVALENCE (NERTS(1),NNERTS) & 1ST DIOIT OF SCENE * * LANDSAT *
C
C PROCEDURE
C -----
C      CALL TRACE
C
C CHECK LANDSAT NUMBER
C
C      IF( (NERTS(1).LT.1).OR.(NERTS(1).GT.6) ) GO TO 600
C
C      LSAT-1    LSAT-2    LSAT-3    LSAT-4
C      00 TO: 100.    200.                    & 0000 TO 0999 DAYS SINCE LAUNCH

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

OCERT  
002

0 300. 400. 0 0000 TO 9999 DAYS SINCE LAUNCH  
0 100. 200 1. 0 1970 TO 1999 DAYS SINCE LAUNCH  
0 NHERTS

C  
C  
C ASSIGN ARBITRARY CONSTANTS  
C

100 CONTINUE 0 LANDSAT-1  
IF( NERSAT(1).EQ.'1' ) NERSAT(1)='LSAT-1'  
120 IF( NERLIN.EQ.0 ) NERLIN=2340 0 NOMINAL LINES/SCENE  
IF( NERSAM.EQ.0 ) NERSAM=3204 0 NOMINAL SAMPLES/SCENE  
IF( ALTKM.EQ.0 ) ALTKM=912. 0 NOMINAL ALTITUDE (KILOMETRES)  
IF( ALTSAM.EQ.0 ) ALTSAM=ALTKM/.056 0 NOMINAL ALTITUDE (SAMPLES)  
IF( NERCHA.EQ.0 ) NERCHA = 4 0 NOMINAL NUMBER OF CHANNELS  
SCNA=.304045 0 MIRROR SCAN COEF (RADIAN)  
SCNB=0.0-.251320 0 MIRROR SCAN COEF (RADIAN)  
SCNC=.095505 0 MIRROR SCAN COEF (RADIAN)  
SCND=.16.5240 0 MIRROR RADIANT NATURAL FREQ (RAD/SEC)  
SCNTHS=.201490 0 ACTIVE SCAN ANGLE (RADIAN)  
SCNTIS=.032130 0 ACTIVE SCAN PERIOD (SEC)  
00 TO 900

C  
C  
C  
200 CONTINUE 0 LANDSAT-2  
IF( NERSAT(1).EQ.'1' ) NERSAT(1)='LSAT-2'  
IF( NERLIN.EQ.0 ) NERLIN=2340 0 NOMINAL LINES/SCENE  
IF( NERSAM.EQ.0 ) NERSAM=3204 0 NOMINAL SAMPLES/SCENE  
IF( ALTKM.EQ.0 ) ALTKM=912. 0 NOMINAL ALTITUDE (KILOMETRES)  
IF( ALTSAM.EQ.0 ) ALTSAM=ALTKM/.056 0 NOMINAL ALTITUDE (SAMPLES)  
IF( NERCHA.EQ.0 ) NERCHA = 4 0 NOMINAL NUMBER OF CHANNELS  
SCNA=.305540 0 MIRROR SCAN COEF (RADIAN)  
SCNB=0.0-.267250 0 MIRROR SCAN COEF (RADIAN)  
SCNC=.097500 0 MIRROR SCAN COEF (RADIAN)  
SCND=.17.0903 0 MIRROR RADIANT NATURAL FREQ (RAD/SEC)  
SCNTHS=.201500 0 ACTIVE SCAN ANGLE (RADIAN)  
SCNTIS=.032330 0 ACTIVE SCAN PERIOD (SEC)  
00 TO 900

C  
C  
C TEMP CONSTANTS USED FOR TESTING  
C

300 CONTINUE 0 LANDSAT-3  
IF( NERSAT(1).EQ.'1' ) NERSAT(1)='LSAT-3'  
IF( NERLIN.EQ.0 ) NERLIN=2340 0 NOMINAL LINES/SCENE  
IF( NERSAM.EQ.0 ) NERSAM=3204 0 NOMINAL SAMPLES/SCENE  
IF( ALTKM.EQ.0 ) ALTKM=912. 0 NOMINAL ALTITUDE (KILOMETRES)  
IF( ALTSAM.EQ.0 ) ALTSAM=ALTKM/.056 0 NOMINAL ALTITUDE (SAMPLES)  
IF( NERCHA.EQ.0 ) NERCHA = 5 0 NOMINAL NUMBER OF CHANNELS  
SCNA=.305540 0 MIRROR SCAN COEF (RADIAN)  
SCNB=0.0-.267250 0 MIRROR SCAN COEF (RADIAN)  
SCNC=.097500 0 MIRROR SCAN COEF (RADIAN)  
SCND=.17.0903 0 MIRROR RADIANT NATURAL FREQ (RAD/SEC)  
SCNTHS=.201500 0 ACTIVE SCAN ANGLE (RADIAN)  
SCNTIS=.032330 0 ACTIVE SCAN PERIOD (SEC)  
00 TO 900

C  
400 CONTINUE 0 LANDSAT-4

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

GCERT  
003

```
C
C
C FLAG BAD LANDSAT NUMBER
C
000 CALL MODFATL('CONSTANTS AVAILABLE ONLY FOR LSAT-1,2,3')
  NERSAT(1)='LSAT-1'
  NERSAT(2)=' '
  GO TO 120      & ASSIGN CONSTANTS FOR LANDSAT-1 ANYWAY
C
C
C COMPUTE REMAINING CONSTANTS
C
000 NERSEN='MSS'
  IF(CTRLIN.EQ.0) CTRLIN=.5*(NERLIN+1)+0.    & SCENE CENTER
  IF(CTRSAM.EQ.0) CTRSAM=.5*(NERSAM+1)+0.    & SCENE CENTER
  ROLRAC=ROLDEO*RADEO*SCNC
  SCNTM2=SCNTM5/2.
  SCNTIN=(SCNTIS*SCNH)/FLOAT(NERSAM+1)
C
C
C RETURN
C
END
```

SUBROUTINE GCHOM 8 GEOMETRIC CONSTANTS TO REGISTER NON PROJECTION DATA

```

C
C
C HISTORY
C -----
C      % H SCHLOSSER      LEC      '10/79      ORIGINAL CODE
C
C METHOD
C -----
C      ACCORDING TO NERTS(1) ASSIGN/COMPUTE APPROPRIATE CONSTANTS.
C
C EXTERNAL REFERENCES
C -----
C      M0FATL      8 PRINT/COUNT/LOG 'FATAL ERROR' DIAGNOSTIC MESSAGE
C
C EXCEPTIONS
C -----
C      1. IF THE ENTRY POINT THAT WAS CALLED DOES NOT AGREE WITH THE CONTENTS
C          OF NERGEO. THEN A FATAL ERROR IS GENERATED.
C
C      2. IF NERTS(1) IS INVALID THEN SET CONSTANTS TO LANDSAT-1 VALUES AND
C          GENERATE FATAL ERROR.
C
C GLOBAL DECLARATIONS
C -----
C      INCLUDE KOMNER.LIST      8 COMMON ERTS SCENE PARAMETERS
C      INCLUDE KONFIT.LIST      8 COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C
C LOCAL DECLARATIONS
C -----
C      REAL RADEG/.0174532921/      8 RADIANS PER DEGREE
C      EQUIVALENCE (NERTS(1),NNERTS)      8 TO USE ERTS NUMBER AS CASE VARIABLE
C
C PROCEDURE
C -----
C      CALL TRACE('GCHOM')
C      IF(NERGEO.NE.'NON') CALL M0FATL( 'GEOMETRY NOT NON IN GCHOM')
C      GO TO 100
C
C
C
C

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

GCNOM  
002

ENTRY GCLCC & GEOMETRIC CONSTANTS TO REGISTER LCC PROJECTION DATA

CALL TRACE('GCLCC')  
IF(NERGEO.NE.'LCC') CALL MDFATL( 'GEOMETRY NOT LCC IN GCLCC')  
00 TO 100

ENTRY GCPS & GEOMETRIC CONSTANTS TO REGISTER PS PROJECTION DATA

CALL TRACE('GCPS')  
IF(NERGEO.NE.'PS') CALL MDFATL( 'GEOMETRY NOT PS IN GCPS')  
00 TO 100

ENTRY GCSOM & GEOMETRIC CONSTANTS TO REGISTER SOM PROJECTION DATA

CALL TRACE('GCSOM')  
IF(NERGEO.NE.'SOM') CALL MDFATL( 'GEOMETRY NOT SOM IN GCSOM')  
00 TO 100

ENTRY GCUTH & GEOMETRIC CONSTANTS TO REGISTER UTM PROJECTION DATA

CALL TRACE('GCUTH')  
IF(NERGEO.NE.'UTH') CALL MDFATL( 'GEOMETRY NOT UTM IN GCUTH')

CHECK LANDSAT NUMBER

100 IF( (NERTS(1).LT.1).OR.(NERTS(1).GT.8) ) 00 TO 800

	LSAT-1	LSAT-2	LSAT-3	LSAT-4	
00 TO	110.	220.			& 0000 TO 0999 DAYS SINCE LAUNCH
&			330.	440.	& 0000 TO 9999 DAYS SINCE LAUNCH
&	110.	220			& 1000 TO 1999 DAYS SINCE LAUNCH
& NNERTS					

ASSIGN ARBITRARY CONSTANTS

110 CONTINUE & LANDSAT-1  
IF(NERSAT(1).EQ.' ') NERSAT(1)='LSAT-1'  
IF(NERCHA.EQ.0) NERCHA = 4 & NOMINAL NUMBER OF CHANNELS  
00 TO 900

220 CONTINUE & LANDSAT-2

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

GCHON  
063

```

      IF(NERSAT(1).EQ.' ') NERSAT(1)='LSAT-2'
      IF(MERCHA.EQ.0) MERCHA = 4      & NOMINAL NUMBER OF CHANNELS
      GO TO 900
C
330  CONTINUE      & LANDSAT-3
      IF(NERSAT(1).EQ.' ') NERSAT(1)='LSAT-3'
      IF(MERCHA.EQ.0) MERCHA = 5      & NOMINAL NUMBER OF CHANNELS
      GO TO 900
C
440  CONTINUE      & LANDSAT-4
C
C
C FLAG BAD LANDSAT NUMBER
C
800  CALL MODATL('CONSTANTS AVAILABLE ONLY FOR LANDSAT-1,2,3')
      NERSAT(1)=' '
      NERSAT(2)=' '
      GO TO 110      & ASSIGN CONSTANTS FOR LANDSAT-1 ANYWAY
C
C
C COMPUTE REMAINING CONSTANTS
C
900  IF(MERLIN.EQ.0) MERLIN=2983      & NOMINAL LINES/SCENE
      IF(CTRLIN.EQ.0) CTRLIN=.5*(MERLIN+1)+0.      & SCENE CENTER LINE
      IF(MERSAM.EQ.0) MERSAM=3548      & NOMINAL SAMPLES/SCENE
      IF(CTRSAM.EQ.0) CTRSAM=.5*(MERSAM+1)+79.      & SCENE CENTER SAMPLE
      IF(ALTSAM.EQ.0) ALTSAM=1.0E+7      & PSEUDO ALTITUDE (SAMPLES)
      IF(ALTSM.EQ.0) ALTSM=ALTSAM*.056      & PSEUDO ALTITUDE (KILOMETRES)
      SCNA=1.0      &
      SCNB=0.5E-7      &
      SCNC=0.0      &
      SCND=MERSAM      &
      SCNTHS=2.0*ATAN(0.5*SCND/ALTSAM)      & ACTIVE SCAN ANGLE (RADIAN)
      SCNTIS=1.0E-7      &
      NERSEN='MSS'
      ROLRAC=ROLOEO*RADEO*SCNC
      SCNTH2=SCNTHS/2.      &
      SCNTIM=SCNTIS      &
C
C
C RETURN
C
END

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**OCONST  
001**

```

SUBROUTINE OCONST  & ASSIGNS UNCORRECTED/ORDINATE COORDINATE CONSTANTS
-----
C
C
C HISTORY
C -----
C
C      J C CRISP      LEC      08/15/79      REQUIREMENTS
C      J C CRISP      LEC      09/25/79      ALGORITHM DESIGN
C      J C CRISP      LEC      09/25/79      ALGORITHM CODING
C
C
C METHOD
C -----
C
C      CHECK NERT(1). IF INVALID SET NERTS(1) = 1 ISSUE FATAL MESSAGE.
C      CHECK NERGE0 IF INVALID SET NERGE0 = 'ERT' AND ISSUE FATAL MESSAGE
C      CALL APPROPRIATE OCXXX ACCORDING TO PROJECTION USED.
C
C
C EXTERNAL REFERENCES
C -----
C
C      OCERT      & ASSIGN/COMPUTE CONSTANTS FOR ERTS PROJECTION
C      OCHOM      & ASSIGN/COMPUTE CONSTANTS FOR HOM PROJECTION
C      OCLCC      & ASSIGN/COMPUTE CONSTANTS FOR LAMBERT PROJECTION
C      OCPS       & ASSIGN/COMPUTE CONSTANTS FOR PS PROJECTION
C      OCSOM      & ASSIGN/COMPUTE CONSTANTS FOR SOM PROJECTION
C      OCUTH      & ASSIGN/COMPUTE CONSTANTS FOR UTM PROJECTION
C      MCFATL     & PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C
C
C EXCEPTIONS
C -----
C
C      1. THE FOLLOWING CONDITIONS GENERATE FATAL ERRORS:
C          NERGE0 = 'BAD'
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES.COUNTERS
C      INCLUDE KOMNER.LIST      & ERTS SCENE PARAMETERS
C
C
C LOCAL DECLARATIONS
C -----
C
C      NONE.
C
C
C PROCEDURE
C -----
C
C      CALL TRACE

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

CONST  
002

```
C
C
C CHECK FOR VALID IMAGE DATA GEOMETRY
C
  100 IF(NERGEO.NE.'BAD') GO TO 200
      NERGEO = 'ERT'
      CALL M0FATL('INVALID IMAGE DATA GEOMETRY')
C
C
C CALL APPROPRIATE ROUTINE FOR IMAGE DATA GEOMETRY
C
  200 IF(NERGEO.EQ.'LCC') CALL GCLCC
      IF(NERGEO.EQ.'PS ') CALL GCPS
      IF(NERGEO.EQ.'SON') CALL GCSOM
      IF(NERGEO.EQ.'UTH') CALL GCUTH
      IF(NERGEO.EQ.'HOM') CALL GCHOM
C
      IF(NERGEO.EQ.'ERT') CALL GCERT
C
C
  900 RETURN
C
  END
```



```

SUBROUTINE GENTIC( 8 GENERATE TICKS FOR OUTPUT WINDOW
1 LUNTC)          8 OUTPUT UNIT FOR TICK LISTING (NONE IF 0)
-----
C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      10/05/73      ORIGINAL CODE
C      E H SCHLOSSER      LEC      10/29/79      UPGRADE DOCUMENTATION
C
C
C METHOD
C -----
C
C      COMPUTE TICK COORDINATES IN SPECIFIED COORDINATE SYSTEM. TRANSFORM
C      TO OTHER COORDINATE SYSTEMS. STORE IN TICK TABLE. AND PRINT IF NON-ZERO
C      OUTPUT UNIT SPECIFIED.  SORT TICK TABLE.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      ASSUMES 38-BIT WORD FOR PACKING TICKS INTO TICK TABLE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      MDNOTE      8 PRINT/COUNT/LOG 'NOTE' DIAGNOSTIC MESSAGE
C      MDWARN      8 PRINT/COUNT/LOG 'WARNING' DIAGNOSTIC MESSAGE
C      Q4A          8 GEOGRAPHIC COORDINATES FOR ADJUSTED SCAN COORDINATES
C      A4B          8 ADJUSTED SCAN COORDINATES FOR GEOGRAPHIC COORDINATES
C      P4A          8 PPD COORDINATES FOR ADJUSTED SCAN COORDINATES
C      Q4U          8 GEOGRAPHIC COORDINATES FOR UTM COORDINATES
C      ISRTBA       8 INTEGER BUBBLE SORT ASCENDING
C      RL2ISX       8 REAL TO INTEGER SEXAGENARY ARRAY
C      MDUNIT       8 PRINT PAGE HEADING ON SPECIFIED UNIT
C      DMPTIC       8 DUMP CONTENTS OF TICK TABLE
C
C
C EXCEPTIONS
C -----
C
C      1. IF THE COORDINATE SYSTEM FOR EITHER PRIMARY OR SECONDARY TICKS
C          IS NOT DEFINED, A WARNING DIAGNOSTIC WILL BE ISSUED.
C
C      2. IF THE TICK TABLE IS NOT LARGE ENOUGH TO CONTAIN THE APPROXIMATE
C          NUMBER OF PRIMARY TICKS TO BE GENERATED. A WARNING DIAGNOSTIC IS
C          ISSUED.
C
C      3. IF SPACE IN THE TICK TABLE IS EXHAUSTED WHILE THE TICKS ARE BEING
C          GENERATED. A NOTE DIAGNOSTIC IS GENERATED AND THE REMAINING TICKS
C          ARE GENERATED AND LISTED. BUT NOT STORED IN THE TICK TABLE.
C
C

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

GENTIC  
002

C GLOBAL DECLARATIONS

C -----

C

INCLUDE KONXQT.LIST	% COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
INCLUDE KONNER.LIST	% COMMON ERTS SCENE PARAMETERS
INCLUDE KONFIT.LIST	% COMMON ADJUSTMENT/REGISTRATION PARAMETERS
INCLUDE KOMOHV.LIST	% COMMON OUTPUT WINDOW PACKETS
INCLUDE KOMTBL.LIST	% COMMON MULTI-PURPOSE TABLE
INCLUDE WINDEF.LIST	% DEFINE STRUCTURE OF WINDOW PACKETS

C

C

C LOCAL DECLARATIONS

C -----

C

LOGICAL THERID	% IS TRANSVERSE MERCATOR CENTRAL MERIDIAN SPECIFIED?
REAL ZONE	% ZONE COMPUTED FROM UTMCHD (ONLY VALID IF INTEGER)
INTEGER NZONE	% VALID UTM ZONE OR -9999

C

C

C PROCEDURE

C -----

C

CALL TRACE

C

C

C INITIALIZE

C

KTBLTY='TICK'  
NTICKS=0  
NLINE=100 % TO FORCE PRINTING OF HEADING  
IF(LUNTIC.LT.0) GO TO 760 % DON'T GENERATE TICKS  
MUNIT=LUNTIC  
IF(INCTLPT.LT.5) MUNIT=0 % DON'T LIST TICKS BASED ON NOMINAL CONTROL

C

C

C START WITH PRIMARY TICKS

C

LVL TIC=0  
NOD TIC=WTIC

C

C

C MARK UTM TICK COORDINATES UNKNOWN

C

200 UTMN=9E+32  
UTMH=9E+32

C

C

C CHECK IF UTM ZONE OR CENTRAL MERIDIAN HAS BEEN SPECIFIED

C

THERID=.FALSE.  
IF(ABS(UTMCHD).LT.180.) THERID=.TRUE.  
NZONE=-9999  
ZONE=(183.-UTMCHD)/6.  
IF(AINT(ZONE).EQ.ZONE) NZONE=ZONE

C

C

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

GEN TIC  
003

C FIND WINDOW TYPE

```
C
      IF((KSYOWH(NOOTIC).EQ.'SCA') GO TO 300
      IF((KSYOWH(NOOTIC).EQ.'DEO').OR.
&      (KSYOWH(NOOTIC).EQ.'MIN')) GO TO 400
      IF((KSYOWH(NOOTIC).EQ.'KH ').OR.
&      (KSYOWH(NOOTIC).EQ.'MET')) GO TO 600
      CALL MDHARN('NO TICKS')
      GO TO 900
```

C

C

C CONVERT ENVELOPE FROM SCANNER UNITS TO TICK UNITS

C

```
300 MINTNS=(MSAOWH(WLIN.WMIN)+MSAOWH(WLIN.NOOTIC)-1)/
&      MSAOWH(WLIN.NOOTIC)
      MAXTNS=MSAOWH(WLIN.WMAX)/MSAOWH(WLIN.NOOTIC)
      MINTEN=(MSAOWH(WSAM.WMIN)+MSAOWH(WSAM.NOOTIC)-1)/
&      MSAOWH(WSAM.NOOTIC)
      MAXTEN=MSAOWH(WSAM.WMAX)/MSAOWH(WSAM.NOOTIC)
      CALL CHKTIC($760)
```

C

C

C GENERATE SCANNER TICKS

C

```
      DO 360 MTICNS=MINTNS.MAXTNS
          ADJLIN=MTICNS*MSAOWH(WLIN.NOOTIC)
          DO 330 MTICEW=MINTEN.MAXTEN
              ADJSAM=MTICEW*MSAOWH(WSAM.NOOTIC)
              CALL 040(GEOLAT.GEOLON. ADJLIN.ADJSAM)
              IF(ITERID) CALL 040(UTME.UTMN. GEOLAT.GEOLON.UTMCMD)
              CALL TICPAK
          330 CONTINUE
      360 CONTINUE
      GO TO 700
```

C

C

C CONVERT ENVELOPE FROM DEGREES TO TICK UNITS

C

```
400 MINTNS=GEDOWH(WLAT.WMIN)/GEDOWH(WLAT.NOOTIC)+.999
      MAXTNS=GEDOWH(WLAT.WMAX)/GEDOWH(WLAT.NOOTIC)+.001
      MINTEN=GEDOWH(WLON.WMIN)/GEDOWH(WLON.NOOTIC)+.999
      MAXTEN=GEDOWH(WLON.WMAX)/GEDOWH(WLON.NOOTIC)+.001
      CALL CHKTIC($760)
```

C

C

C GENERATE GEOGRAPHIC TICKS

C

```
      DO 460 MTICNS=MAXTNS.MINTNS.-1      & TOP TO BOTTOM. SAME AS SCANNER
          GEOLAT=MTICNS*GEDOWH(WLAT.NOOTIC)
          DO 430 MTICEW=MAXTEN.MINTEN.-1      & LEFT TO RIGHT. SAME AS SCANNER
              GEOLON=MTICEW*GEDOWH(WLON.NOOTIC)
              CALL 440(ADJLIN.ADJSAM. GEOLAT.GEOLON)
              IF(ITERID) CALL 040(UTME.UTMN. GEOLAT.GEOLON.UTMCMD)
              CALL TICPAK
          430 CONTINUE
      460 CONTINUE
```

**DAM PACKAGE APPENDIX M  
UTILITY ROUTINES**

**GENTIC  
084**

**00 TO 700**

**C  
C  
C**

**CONVERT ENVELOPE FROM METRES TO TICK UNITS**

**800 MINTEN=UTMOWH(WEA.WMIN)/UTMOWH(WEA.NOOTIC)\*.999  
MAXTEN=UTMOWH(WEA.WMAX)/UTMOWH(WEA.NOOTIC)\*.001  
MINTNS=UTMOWH(WNO.WMIN)/UTMOWH(WNO.NOOTIC)\*.999  
MAXTNS=UTMOWH(WNO.WMAX)/UTMOWH(WNO.NOOTIC)\*.001  
CALL CHKTC(8780)  
IF(THRID) GO TO 840  
CALL MONOTE('ZONE/MERIDIAN NOT DEFINED -- NO UTM TICKS')  
00 TO 700**

**C  
C  
C**

**GENERATE UTM TICKS**

**840 DO 860 MTICNS=MAXTNS.MINTNS.-1      8 TOP TO BOTTOM. SAME AS SCANNER  
UTMN=MTICNS\*UTMOWH(WNO.NOOTIC)  
DO 830 MTICEW=MINTEN.MAXTEN  
UTME=MTICEW\*UTMOWH(WEA.NOOTIC)  
CALL 04U(OEDLAT.OEDLON.      UTME.UTMN.UTMCHD)  
CALL A40(ADJLIN.ADJSAM.      OEDLAT.OEDLON)  
CALL TICPAK**

**830      CONTINUE  
860 CONTINUE**

**C  
C  
C**

**CONTINUE WITH SECONDARY TICKS**

**700 IF(LVLTIC.NE.0) 00 TO 740  
LVLTIC=1  
NOOTIC=MTIC+1  
00 TO 200**

**C  
C  
C**

**CHECK FOR TICK TABLE OVERFLOW**

**740 IF(LVLTIC.NE.2) 00 TO 760  
CALL MONOTE'  
8 'TOO MANY TICKS -- REMAINING TICKS IGNORED'**

**C  
C  
C**

**MARK END OF TICK TABLE AND SORT IT**

**760 KTABLE(NTICKS+1)=99999\*2+18+99999\*2+1      8 MAX LINE. MAX SAMPLE  
CALL ISRTBA(KTABLE.NTICKS)**

**C  
C  
C**

**DUMP TICK TABLE (ONLY IF TRACING)**

**IF(MDUMP.NE.0) CALL DMPTIC**

**C  
C  
C**

**NORMAL RETURN**

DAN PACKAGE APPENDIX H  
UTILITY ROUTINES

GEN TIC  
000

000 RETURN

C  
C  
C  
C  
C  
C

INTERNAL  
SUBROUTINE CNKTIC(S)

IF(

& ((MAXTNS-MINTNS)\*(MAXTEM-MINTEM)/(LVL TIC+1).GE.(KTBLSZ-NTICKS)) & NO ROOM  
& .AND.  
& ((NTICKS\*MBATCH).EQ.0)) & ON PRIMARY TICKS OR IN DEMAND RUN  
& CALL MOWARN('TOO MANY TICKS')  
IF(NDTOTL.NE.0) RETURN 1  
RETURN

C  
C  
C  
C  
C  
C  
C

INTERNAL  
SUBROUTINE TICPAK & PACK TICK COORDINATES INTO TABLE & PRINT THEM

C

DIMENSION JSYTIC(3),LATOMS(3),LONDMS(3)  
DATA JSYTIC /' ','+', ' ' / & PRIMARY, SECONDARY, OMITTED

C  
C  
C

STORE TICK COORDINATES AND LEVEL CODE IN TICK TABLE

CALL P4A(PPDLIN,PPDCOL, ADJLIN,ADJSAH)  
IF(

& (PPDLIN.LT.PPDOWH(HLIN.WMIN)).OR.  
& (PPDLIN.GT.PPDOWH(HLIN.WMAX)).OR.  
& (PPDCOL.LT.PPDOWH(HSAH.WMIN)).OR.  
& (PPDCOL.GT.PPDOWH(HSAH.WMAX))) GO TO 900  
IF(1RFD.NE.0) GO TO 110 & TRUNCATE FOR SCALED OUTPUT  
PPDLIN=PPDLIN+.5  
PPDCOL=PPDCOL+.5

110 IPLIN=PPDLIN & TRUNCATE  
IPCOL=PPDCOL & TRUNCATE  
IF(NTICKS.LT.KTBLSZ-1) GO TO 120  
LVL TIC=2 & NO ROOM IN TABLE -- SYMBOL OMITTED  
GO TO 300

120 NTICKS=NTICKS+1  
KTABLE(NTICKS)=

& IPLIN\*2\*\*18+ & BITS 18 THRU 35 (F'D(00.18. .. ))  
& IPCOL\*2+ & BITS 01 THRU 17 (F'D(18.17. .. ))  
& LVL TIC & BIT 00 (F'D(35.01. .. ))

C  
C  
C  
C

PRINT TICK SYMBOL AND COORDINATES

300 IF(NUNIT.EQ.0) GO TO 900  
IF(NDTOTL.NE.0) GO TO 900

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

02TIC  
000

```

      IF(NLINE.GT.LPAGE) CALL TICHD0(NZONE,UTMCHD)
      CALL RL2ISX(0EDLAT+.00001,LATDMS,3,RLTEMP)
      CALL RL2ISX(0EDLON+.00001,LONDMS,3,RLTEMP)
      NSALIN=ADJLIN      3 TRUNCATE
      NSASAM=ADJSAM      3 TRUNCATE
      WRITE(NUNIT,325)
      1 JSYTIC(LVLTIC+1).
C     2 NSALIN,NSASAM.
      2 ADJLIN,ADJSAM.
      3 LATDMS,LONDMS.
      9 0EDLAT,0EDLON.
      6 UTMN,UTMN.
      7 IPLIN,IPCOL
      325 FORMAT(
      1 3X,A2.
      2 1X,2F9.2.
      3 1X,14,'.J2.'.J2.
      4 1X,14,'.J2.'.J2.
      5 2X,2F10.4.
      6 2X,-3P,2F9.3,0P.
      7 2X,217)
      NLINE=NLINE+1
      900 RETURN

```

C  
C  
C  
C  
C  
C  
C

```

      INTERNAL
      SUBROUTINE TICHD0(NZONE,UTMCHD)
      CALL MDUNIT(4,NUNIT)
      IF(NZONE.LT.0) WRITE(NUNIT,115) UTMCHD
      115 FORMAT(''.6X,'MERIDIAN '.F9.4)
      WRITE(NUNIT,125) NZONE
      125 FORMAT(
      1 ' TICK          SCANNER'.4X.
      3 ' (00:MM:SEC)'.6X.
      5 ' (DEGREES)'.6X.
      6 ' UTM ZONE '.12,' (KM)'.2X.
      7 ' PRINTER')
      WRITE(NUNIT,145)
      145 FORMAT(
      1 ' SYMBOL    LINE SAMPLE '.
      3 ' LATITUDE  LONGITUDE '.
      5 ' LATITUDE  LONGITUDE '.
      6 ' EASTING  NORTHING '.
      7 ' LINE COLUMN')
      RETURN

```

C  
C

END

RECEIVED 05/21/65  
OF THE QUALITY

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

GETDSR  
001

```

SUBROUTINE GETDSR( I GET DISK SYMBOLIC RECORD
O INSTAT. I INPUT STATUS:      . . . NORMAL COMPLETION
                                'FS'  FILE SEPARATOR (SEOF)
                                'EOF'  END OF FILE
                                'OFL'  BUFFER OVERFLOW ( LENGTH > 120 )
O IMAGE.  I IMAGE BUFFER (MAX 120 CHARS)
O LENGTH. I LENGTH OF IMAGE IN CHARS. INCLUDES TRAILING BLANKS. IF PRESENT
O LOCXT.  I LOCATION OF NEXT RECORD WITHIN FILE ( < 0 = END-OF-FILE )
I LOCREC. I LOCATION OF RECORD WITHIN FILE
I NAMEFIL I NAME OF FILE
            I NAME MUST CONTAIN 12 CHARACTERS, INCLUDING TRAILING BLANKS
            -----
C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      31/22/79      ORIGINAL REQUIREMENTS
C
C
C METHOD
C -----
C
C      TO BE DETERMINED.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      UNIVAC EXEC B PROGRAM FILE NAMING CONVENTIONS AND EXECUTIVE REQUESTS.
C
C EXTERNAL REFERENCES
C -----
C
C      MOVEST      I MOVE CHARACTER STRING
C      ERION       I SUBMIT EXEC-B FACILITY REQUEST CONTROL STATEMENT
C
C EXCEPTIONS
C -----
C
C      1. FILE OR ELEMENT DOES NOT EXIST.
C
C      2. FILE OR ELEMENT IS NOT SYMBOLIC.
C
C      3. FILE NOT ASSIGNED TO CURRENT RUN.
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KONXGT.LIST
C
C

```

**DAN PACKAGE APPENDIX H  
UTILITY ROUTINES**

**OETDSR  
002**

**C LOCAL DECLARATIONS**

**C -----**

**C**

**INTEGER NANFIL(2)      8 ARGUMENT**

**C**

**C**

**C PROCEDURE**

**C -----**

**C**

**C**

**CALL TRACE**

**C**

**CALL ERPRNT(1,5,'\*\*OETDSR NOT YET IMPLEMENTED\*\*')**

**C**

**C**

**960 RETURN**

**END**





DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

GETRAD  
002

```

C
C
C      2. IF IRRECOVERABLE ERROR OCCURRED DURING READ. RETURN IS
C      IMMEDIATE WITH NO VALID DATA.
C
C      3. IF COMPUTED VALUE FOR TRANSFORMED CHANNELS IS LESS THAN ZERO OR
C      GREATER THAN HIGHEST INFO VALUE (FROM PREAMBLE). THEN THE VALUE
C      WILL BE SET TO THE NEAREST EXTREME.
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES. COUNTERS
C      INCLUDE KOMNER.LIST      & COMMON ERTS SCENE PARAMETERS
C      INCLUDE KOMKLS.LIST      & COMMON CLASSIFICATION INFO
C      INCLUDE KOMIRT.LIST      & COMMON IRRADIANCE TRANSFORMATION COEFFICIENTS
C      INCLUDE PXBDEF.LIST      & DEFINE STRUCTURE OF PIXEL BUFFER
C      INCLUDE MAXINT.LIST      & DEFINE MAXIMUM INTEGER
C
C
C LOCAL DECLARATIONS
C -----
C
C      PARAMETER INFMAX=127      & MAX INFO VALUE FOR TRANSFORMED CHANNEL
C      PARAMETER NOINFL=128      & NO INFO FLAG FOR TRANSFORMED CHANNEL
C      PARAMETER NODATH=129      & NO DATA THRESHOLD FOR TRANSFORMED CHANNEL
C
C      INTEGER MPXBUF(NHIBF,NBUFS) & ARGUMENT
C      LOGICAL SHRPEN             & TRUE - SHARPEN CHANNEL
C                                  & FALSE - DO NOT SHARPEN
C
C      INTEGER IBINTY             & BIN TYPE
C      INTEGER JSTAT              & TEMPORARY READ3 STATUS
C      INTEGER LTQUAL(2)          & LINEAR TRANSFORM DATA QUALITY
C      INTEGER LTLSAM(2)          & LINEAR TRANSFORM LOW SAMPLE
C      INTEGER LTHSAM(2)          & LINEAR TRANSFORM HIGH SAMPLE
C      INTEGER LTLBIN(2)          & LINEAR TRANSFORM LOW BIN
C      INTEGER LTHBIN(2)          & LINEAR TRANSFORM HIGH BIN
C      INTEGER NBFAC1             & NUMBER OF BUFFER FOR LINEAR TRANSFORM 1
C      INTEGER NBFAC2             & NUMBER OF BUFFER FOR LINEAR TRANSFORM 2
C      INTEGER IPOL1              & POLAR TRANSFORM CHANNEL 1
C      INTEGER IPOL2              & POLAR TRANSFORM CHANNEL 2
C      INTEGER LRT12(2)           & LINEAR TRANSFORM CHANNELS * 2**12
C
C      DEFINE ITAN12=(LRT12(2)*2**12)/LRT12(1)      & TANGENT*2**12
C      DEFINE ICOT12=(LRT12(1)*2**12)/LRT12(2)      & COTAN*2**12
C
C      RADIUS=(LRT(1)**2+LRT(2)**2)**.5
C      = LRT(1) * LRT(2) * TAN(THETA) * .4142      & APPROX.
C      DEFINE LTAN12=(LRT12(1)*(LRT12(2)*ITAN12)/9889) & LENGTH OF RADIUS*2**12
C      DEFINE LCOT12=(LRT12(2)*(LRT12(1)*ICOT12)/9889) & LENGTH OF RADIUS*2**12
C
C      ARCTAN= (3.880894*TAN-TAN**2) /3.88089      & APPROX.
C      DEFINE KTAN12=( (15.796*ITAN12-ITAN12*ITAN12)/15024) & ARCTAN*2**12
C      ARCCOT=1.57080-(3.880894*COT-COT**2) /3.88089      & APPROX.
C      DEFINE KCOT12=(6434-(15896*ICOT12-ICOT12*ICOT12)/15024) & ARCCOT*2**12
C

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

GETRAD  
003

C THE ABOVE ARCTAN AND ARCCOTAN APPROXIMATIONS ARE DESIGNED FOR USE WITHIN  
C THEIR RESPECTIVE HALVES OF THE FIRST QUADRANT. THE FOLLOWING TABLE SHOWS  
C THAT THEIR MAXIMUM ERROR WHEN SO RESTRICTED WITHIN THE FIRST QUADRANT IS  
C  $\pm 0.21$  DEGREES, BUT THAT THE ERROR RAPIDLY INCREASES (WITHOUT SOUND) IN  
C THE ADJACENT SECOND AND FOURTH QUADRANTS.

C	TAN OR	CORRECT	APPROXIMATE	CORRECT	APPROXIMATE
C	COT	ATAN	ATAN	ACOT	ACOT
C	+1.0000	+45	+45.00	+45	+45.00
C	+0.8391	+40	+39.07	+50	+50.13
C	+0.7002	+35	+34.79	+55	+55.21
C	+0.5773	+30	+29.79	+60	+60.21
C	+0.4663	+25	+24.87	+65	+65.13
C	+0.3840	+20	+20.00	+70	+70.00
C	+0.2879	+15	+15.12	+75	+74.88
C	+0.1763	+10	+10.20	+80	+79.80
C	+0.0875	+5	+5.18	+85	+84.82
C	0.0000	0	0.00	+90	+90.00
C	-0.0875	-5	-5.42	+95	+95.42
C	-0.1763	-10	-11.17	+100	+101.17
C	-0.2879	-15	-17.36	+105	+107.36
C	-0.3840	-20	-24.14	+110	+114.14
C	-0.4663	-25	-31.66	+115	+121.66
C	-0.5773	-30	-40.20	+120	+130.20
C	-0.7002	-35	-50.10	+125	+140.10
C	-0.8391	-40	-61.86	+130	+151.86
C	-1.0000	-45	-76.24	+135	+166.24

C PROCEDURE

C -----

C IF (MOD(MSALIN,50).EQ.0) CALL TRACE (KMR\*IN(MSALIN))

C CHECK IF CHANNEL AVAILABILITY HAS BEEN DETERMINED

C DO 100 NBF=1,5

IF (NBFCHR(NBF).NE.999) GO TO 110

100 CONTINUE

CALL MDATL( 'CHANNEL AVAILABILITY NOT DETERMINED BY CALCHA')

GO TO 900

C READ (AND SHARPEN) ORIGINAL CHANNELS. IF SHARPENING IS INDICATED READ  
C INTO LAST BUFFER AS A WORK BUFFER AND THEN SHARPEN INTO CORRECT DESTI-  
C NATION BUFFER. IF SHARPENING IS NOT INDICATED, READ DIRECTLY INTO  
C DESTINATION BUFFER.

C 110 IBINTY='BYT' & USE 'BYT' AS DEFAULT BIN TYPE

JSTAT=' '

DO 190 NCHR=1,NERCHA

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

GETRAD  
664

```

NBF=NBFCNR(NCHR)
SHRPN=(IRSF12(NCHR,1).NE.0 .OR. IRSF12(NCHR,2).NE.0)
IF (SHRPN .AND. (NBF.GE.NBUF)) GO TO 190
IF ( (.NOT.SHRPN) .AND. (NBF.GT.NBUF)) GO TO 190
NBF10=NBF
IF (SHRPN) NBF10=NBUF
CALL READ3 (MPXBUF(1,NBF10),NMIBF,1,STAT,
            NSALIN,NCHR,NSASLO,NSASH1)
IF (1,STAT.EQ.'EOF' .OR. 1,STAT.EQ.'BADF' .OR. 1,STAT.EQ.'OFL')
1 GO TO 900      3 IRRECOVERABLE ERROR
IF (NMIBF.EQ.0) GO TO 900      3 GETRAD WAS REQUESTED TO POSITION ONLY
IF (1,STAT.EQ.'BADR') GO TO 120      3 NO DATA
IBINTY=MPXBUF(PXBINT,NBF10)
IF (SHRPN) CALL SHASAM(MPXBUF(1,NBF),MPXBUF(1,NBF10),
1 IRSF12(NCHR,1),IRSF12(NCHR,2))
GO TO 190
120 JSTAT='BADR'
IF ( (.NOT. SHRPN) ) GO TO 190
NMIPRE=PXBSIN-1      3 TRANSFER PREAMBLE TO CORRECT DESTINATION
DO 130 I=1,NMIPRE      3 BUFFER SINCE SHASAM NOT CALLED
    MPXBUF(1,NBF)=MPXBUF(1,NBF10)
130 CONTINUE
190 CONTINUE
    ISTAT=JSTAT
C
C
C INITIALIZE LINEAR TRANSFORMED CHANNELS PARAMETERS
C
IF (IRTTYP.EQ.'RAW') GO TO 900      3 NO TRANSFORMATIONS
DO 200 I=1,2
    LTQUAL(I)=0
    LTLSAM(I)=0
    LTNSAM(I)=MAXINT
    LTLOBIN(I)=0
    LTNOBIN(I)=MAXINT
200 CONTINUE
C
C
C FOR EACH LINEAR TRANSFORM COMPUTE THE RANGE OF COMMON SAMPLES
C AND BINS OF THE CHANNELS THAT COMPRISE THE TRANSFORM
C
DO 230 K=1,2
    DO 210 NCHR=1,NERCHA
        IF (LRTW12(NCHR,K).EQ.0.0) GO TO 210
        NBF=NBFCNR(NCHR)
        LTQUAL(K)=MAX0 (LTQUAL(K),MPXBUF(PXQUAL,NBF))
        LTLSAM(K)=MAX0 (LTLSAM(K),MPXBUF(PXLSAM,NBF))
        LTNSAM(K)=MIN0 (LTNSAM(K),MPXBUF(PXNSAM,NBF))
        LTLOBIN(K)=MAX0 (LTLOBIN(K),MPXBUF(PXLOBIN,NBF))
        LTNOBIN(K)=MIN0 (LTNOBIN(K),MPXBUF(PXNOBIN,NBF))
210 CONTINUE
        IF (LTNOBIN(K).EQ.MAXINT) GO TO 220      3 NO TRANSFORM
        IF (LTLOBIN(K).GT.LTNOBIN(K)) GO TO 220
        GO TO 230
220 LTQUAL(K)=0
    LTLSAM(K)=0

```

DAW PACKAGE APPENDIX M  
UTILITY ROUTINES

GETRAB  
009

LTHSAM(K)=0  
LTLBIN(K)=0  
LTHBIN(K)=0

230 CONTINUE

C

C

C SINCE TRANSFORMED CHANNELS REPLACE ORIGINAL BUFFERS 1 AND 2 WHICH  
C MAY CONTAIN CHANNELS NEEDED IN THE TRANSFORMS. THE TRANSFORMED  
C CHANNELS ARE COMPUTED IN PARALLEL SAMPLE BY SAMPLE AND HELD IN  
C TEMPORARY LOCATIONS UNTIL BOTH TRANSFORMED CHANNELS ARE COMPUTED.

C

C

IF (LTQUAL(1).EQ.9 .AND. LTQUAL(2).EQ.9) GO TO 900 3 NO DATA FOR  
3 EITHER CHANNEL

IF (LTQUAL(1).EQ.9) GO TO 240 3 NO DATA FOR TRANSFORMED CHANNEL 1

IF (LTQUAL(2).EQ.9) GO TO 250 3 NO DATA FOR TRANSFORMED CHANNEL 2

LOWBIN=MIN0(LTLBIN(1),LTLBIN(2))

HIBIN=MAX0(LTHBIN(1),LTHBIN(2))

GO TO 260

240 LOWBIN=LTLBIN(2)

HIBIN=LTHBIN(2)

GO TO 260

250 LOWBIN=LTLBIN(1)

HIBIN=LTHBIN(1)

260 NBFAC1=1

NBFAC2=2

IF (LIMCH(1).EQ.1) GO TO 270

NBFAC1=2

NBFAC2=1

270 CONTINUE

C

C

C USE INTERNAL SUBROUTINE TO COMPUTE PIXEL VALUES FOR BINS OF  
C TRANSFORMED CHANNELS

C

IF (IBINTY.EQ.'BYT') CALL TRNFOR(GETBYT,PUTBYT)

IF (IBINTY.EQ.'CHR') CALL TRNFOR(GETICE,PUTICE)

IF (IBINTY.EQ.'INT') CALL TRNFOR(GETINT,PUTINT)

C

C

C MODIFY PREAMBLES TO REFLECT TRANSFORMED CHANNELS

C

MPXBUF(PXCHAN,NBFAC1)=1

MPXBUF(PXCHAN,NBFAC2)=2

MPXBUF(PXNOIN,NBFAC1)=NOINFL

MPXBUF(PXNOIN,NBFAC2)=NOINFL

MPXBUF(PXNODA,NBFAC1)=NODATH

MPXBUF(PXNODA,NBFAC2)=NODATH

IF (IRRTYP.EQ.'POL') GO TO 280

MPXBUF(PXQUAL,NBFAC1)=LTQUAL(1)

3 LINEAR

MPXBUF(PXQUAL,NBFAC2)=LTQUAL(2)

MPXBUF(PXBINT,NBFAC1)=IBINTY

MPXBUF(PXBINT,NBFAC2)=IBINTY

MPXBUF(PXLBIN,NBFAC1)=LTLBIN(1)

MPXBUF(PXLBIN,NBFAC2)=LTLBIN(2)

MPXBUF(PXLSAM,NBFAC1)=LTHSAM(1)

MPXBUF(PXLSAM,NBFAC2)=LTHSAM(2)

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

GETRAD  
000

```

      MPXBUF(PXHBIN,NBFAC1)=LTHBIN(1)
      MPXBUF(PXHBIN,NBFAC2)=LTHBIN(2)
      MPXBUF(PXHSAM,NBFAC1)=LTHSAM(1)
      MPXBUF(PXHSAM,NBFAC2)=LTHSAM(2)
      GO TO 900
200 MPXBUF(PXQUAL,NBFAC1)=MAX0(LTQUAL(1),LTQUAL(2))      & POLAR
      IF (MPXBUF(PXQUAL,NBFAC1).EQ.9) GO TO 300      & POLAR COOR VALID ONLY
C                                          & IF BOTH LIN CHANNELS
C                                          & HAVE DATA
      MPXBUF(PXQUAL,NBFAC2)=MPXBUF(PXQUAL,NBFAC1)
      MPXBUF(PXLBIN,NBFAC1)=MAX0(LTLBIN(1),LTLBIN(2))
      MPXBUF(PXLBIN,NBFAC2)=MPXBUF(PXLBIN,NBFAC1)
      MPXBUF(PXHBIN,NBFAC1)=MIN0(LTHBIN(1),LTHBIN(2))
      MPXBUF(PXHBIN,NBFAC2)=MPXBUF(PXHBIN,NBFAC1)
      MPXBUF(PXLSAM,NBFAC1)=MAX0(LTLSAM(1),LTLSAM(2))
      MPXBUF(PXLSAM,NBFAC2)=MPXBUF(PXLSAM,NBFAC1)
      MPXBUF(PXHSAM,NBFAC1)=MIN0(LTHSAM(1),LTHSAM(2))
      MPXBUF(PXHSAM,NBFAC2)=MPXBUF(PXHSAM,NBFAC1)
      IF (MPXBUF(PXLBIN,NBFAC1).GT.MPXBUF(PXHBIN,NBFAC1)) GO TO 300
      GO TO 900
300 DO 310 K=1,2
      MPXBUF(PXQUAL,K)=9
      MPXBUF(PXLBIN,K)=0
      MPXBUF(PXHBIN,K)=0
      MPXBUF(PXLSAM,K)=0
      MPXBUF(PXHSAM,K)=0
310 CONTINUE
900 RETURN
C
C
C
C
C
C
      INTERNAL
      SUBROUTINE TRNFOR ( & COMPUTE PIXEL VALUES FOR TRANSFORMED CHANNELS
1 GETBIN.      & NAME OF ROUTINE TO GET A BIN  GETBYT/GETICE/GETINT
1 PUTBIN)      & NAME OF ROUTINE TO PUT A BIN  PUTBYT/PUTICE/PUTINT
C
C
C COMPUTE LINEAR TRANSFORMED CHANNELS.  THEN COMPUTE POLAR TRANSFORMED
C CHANNELS IF REQUESTED.
C
      DO 400 I=LOWBIN,HIHBIN
      LRT12(1)=LRTB12(1)      & BIAS FOR LINEAR.1
      LRT12(2)=LRTB12(2)      & BIAS FOR LINEAR.2
      DO 410 NCHR=1,NERCHA
      NBF=NBFCHR(NCHR)
      IF (NBF.GT.5) GO TO 410
      CALL GETBIN (PIX,MPXBUF(PXBINS,NBF),1)
      LRT12(1)=LRT12(1) + LRTW12(NCHR,1)*PIX      & WEIGHT FOR LINEAR.1
      LRT12(2)=LRT12(2) + LRTW12(NCHR,2)*PIX      & WEIGHT FOR LINEAR.2
410 CONTINUE
      IF (IRTTYP.EQ.'POL') GO TO 420

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

GETRAD  
007

C STORE LINEAR TRANSFORMED CHANNELS

C  
LRT12(1)=LRT12(1)/2\*\*12  
LRT12(2)=LRT12(2)/2\*\*12

C  
C  
C LIMIT CHECK AND STORE LINEAR TRANSFORM CHANNELS  
C

LRT12(1)=MAX(0,MING(INFMAX,LRT12(1)))  
LRT12(2)=MAX(0,MING(INFMAX,LRT12(2)))  
CALL PUTBIN (MPXBUF(PXBINS,NBFAC1),1, LRT12(1))  
CALL PUTBIN (MPXBUF(PXBINS,NBFAC2),1, LRT12(2))  
GO TO 480

C  
C  
C POLAR TRANSFORMS FROM TANGENT OF ANGLE BETWEEN LINEAR TRANSFORMED  
C CHANNELS  
C

420 IF (LRT12(2).GT.LRT12(1)) GO TO 440  
IF (LRT12(1).EQ.0) LRT12(1)=2 & PREVENT DIVISION BY ZERO  
IPOL1 = (LTAN12\*NRT012(1)+NRTB24(1))/2\*\*24 & BRIGHTNESS  
IF (LRT12(2).LT.0) GO TO 430  
IPOL2=(KTAN12\*NRT012(2)+NRTB24(2))/2\*\*24  
GO TO 470  
430 LRT12(2)=ABS(LRT12(2))  
IPOL2=(-KTAN12\*NRT012(2)+NRTB24(2))/2\*\*24  
GO TO 470

C  
C  
C POLAR TRANSFORMS FROM COTANGENT OF ANGLE BETWEEN LINEAR  
C TRANSFORMED CHANNELS  
C

440 IF (LRT12(2).EQ.0) LRT12(2)=2  
IPOL1=(LCOT12\*NRT012(1)+NRTB24(1))/2\*\*24  
IF (LRT12(1).LT.0) GO TO 450  
IPOL2=(KCOT12\*NRT012(2)+NRTB24(2))/2\*\*24  
GO TO 470  
450 LRT12(1)=ABS(LRT12(1))  
IPOL2=(16777216-KCOT12\*NRT012(2)+NRTB24(2))/2\*\*24 816777216=180 DEG  
1 & 2\*\*24 IN RADIANS

C  
C  
C LIMIT CHECK AND STORE POLAR CHANNELS  
C

470 IPOL1=MAX(0,MING(INFMAX,IPOL1))  
IPOL2=MAX(0,MING(INFMAX,IPOL2))  
CALL PUTBIN (MPXBUF(PXBINS,NBFAC1),1, IPOL1)  
CALL PUTBIN (MPXBUF(PXBINS,NBFAC2),1, IPOL2)  
480 CONTINUE  
RETURN  
END

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

OETS  
001

SUBROUTINE OETS: 0 GET REMAINING CONTENTS OF UNIT 5 BUFFER  
0 KHALL. 0 RETURN AS CHARACTER STRING (UNCHANGED IF MISSING)  
1 KLEN. 0 LENGTH IN CHARACTERS (WILL BE PADDED WITH BLANKS TO 80 BY)  
1 MSHARN) 0 WARNING MESSAGE IF MISSING (\* MEANS REQUIRED)  
-----

C  
C  
C  
C HISTORY  
C -----

C E M SCHLOSSER LEC 03/08/79 ORIGINAL CODE IN READS  
C E M SCHLOSSER LEC 02/14/79 REVISE & MAKE SEPARATE SUBROUTINE  
C  
C

C METHOD  
C -----

C THIS SUBROUTINE OETS ALL FREE-FORMAT DATA FIELDS FROM UNIT 5 (CARD READER  
C OR DEMAND TERMINAL) FOR THE DAM PACKAGE.  
C  
C

C FREE-FORMAT INPUT RULES  
C -----

1. RECORDS -- AN INPUT RECORD CONTAINS UP TO 80 COLUMNS ENDING WITH  
COLUMN 80 (CARD READER) OR TERMINATED BY A CARRIAGE RETURN (TERMINAL).
2. SENTINELS -- THE FOLLOWING CHARACTERS IN COLUMN 1 DESIGNATE SPECIAL  
RECORDS:
  - 0 CONTROL CARD (IDENTIFIES END-OF-FILE)
  - \* REMARK CARD (IGNORED BUT LISTED ON OUTPUT)
3. DELIMITERS -- ALL OTHER RECORDS CONTAIN DATA FIELDS AND/OR COMMENT  
FIELDS, ORGANIZED INTO BLOCKS, AND SEPARATED BY THE FOLLOWING DELIM-  
ITERS:
  - . ENDS DATA FIELD (NEXT DATA FIELD FOLLOWS)
  - .. ENDS LAST DATA FIELD IN BLOCK (COMMENT FIELD FOLLOWS)
  - ... ENDS LAST FIELD IN BLOCK (NEXT BLOCK FOLLOWS)
4. BLOCKS -- EACH BLOCK IS NORMALLY TERMINATED BY A TRIPLE COMMA (...) OR  
THE END-OF-RECORD, WHICHEVER COMES FIRST.
5. SPANNED BLOCKS -- (ACTIVATED FOR CURRENT BLOCK BY CALL TO SPANS) IF A  
PHYSICAL RECORD ENDS WITH A SINGLE COMMA, THEN THE NEXT FIELD FOR THE  
CURRENT BLOCK IS TAKEN FROM THE NEXT PHYSICAL RECORD.
6. FIELDS -- EACH DATA FIELD IS COMPOSED OF A LEADING, OR SIGNIFICANT,  
PART AND A TRAILING, OR NON-SIGNIFICANT, PART. FOR NUMERIC FIELDS THE  
SIGNIFICANT PART IS TERMINATED BY THE FIRST EMBEDDED BLANK. FOR  
NON-NUMERIC FIELDS THE MAXIMUM LENGTH OF THE SIGNIFICANT PART  
IS SPECIFIED IN THE CALLING ARGUMENT. FOR ALPHABETIC FIELDS  
(ENTRY OETSAL) THE ENTIRE FIELD MUST BE ALPHABETIC. FOR ALL OTHER  
FIELDS THE NON-SIGNIFICANT PART IS IGNORED ENTIRELY.
7. SCALING -- ALL INPUT TO OETS5X, OETSFR, AND OETSRL IS IMMEDIATELY



C MULTIPLIED BY RLCOEF BEFORE BEING PROCESSED FURTHER.  
C  
C  
C B. DIAGNOSTICS -- THE WARNING MESSAGE (MSHARN) IS PRINTED BY GETS. GETSAL.  
C GETSKH. GETSSX. GETSFR. GETSRL. AND GETSIN UNDER THE FOLLOWING  
C CONDITIONS:  
C A.) IF THE FIELD IS INVALID. OR  
C B.) IF THE FIELD IS MISSING AND ALL MISSING FIELDS. UP TO AND  
C INCLUDING THE CURRENT FIELD. ARE REQUIRED FIELDS. (A REQUIRED  
C FIELD IS INDICATED BY A WARNING MESSAGE WHICH STARTS WITH AN  
C ASTERISK.)

ENTRY POINT	LEGAL FORMATS	FUNCTION
GETS		RETURN REMAINING CONTENTS OF UNIT 5 BUFFER
GETSAL	AL	RETURN NEXT DATA FIELD AS CHARACTER STRING
GETSKH	KH SX FR RL IN	RETURN NEXT DATA FIELD AS CHARACTER STRING
GETSSX	SX RL IN	RETURN NEXT DATA FIELD AS REAL NUMBER
GETSFR	FR RL IN	RETURN NEXT DATA FIELD AS REAL NUMBER
GETSRL	RL IN	RETURN NEXT DATA FIELD AS REAL NUMBER
GETSIN	IN	RETURN NEXT DATA FIELD AS INTEGER
UNOETS		BACK UP 1 FIELD (CALL ONLY ONCE IN SUCCESSION)

#### FORMAT TYPES

C AL ALPHABETIC STRING (26 LETTERS PLUS BLANK PLUS CLASS OF FIRST CHAR)  
C KH CHARACTER STRING (EXCLUDING COMMAS)  
C SX DECIMAL-CODED SEXAGENARY NUMBER (EX. -109:37:30.5)  
C SEXAGENARY PLACES ARE SEPARATED BY ':' OR '/'  
C EVERY PLACE BUT THE LEFTMOST MUST BE UNSIGNED & LESS THAN 60  
C ONLY THE RIGHTMOST PLACE MAY CONTAIN A DECIMAL POINT  
C FR FRACTION (EX. -3/8)  
C RL REAL NUMBER  
C IN INTEGER

#### MACHINE-DEPENDENT CODE

C NONE.

#### EXTERNAL REFERENCES

C  
C NEXTOK 3 GET POINTERS TO NEXT TOKEN  
C GETOKH 3 GET CHARACTER STRING DATA FIELD FROM BUFFER  
C SREADS 3 SPANNED READ OF UNIT 5  
C HARNB 3 SUBMIT WARNING FOR INVALID/MISSING FIELD FROM UNIT 5  
C PUTCHR 3 PUT CHARACTER INTO CHARACTER STRING  
C NOVCST 3 MOVE CHARACTER STRING  
C DCODE 3 DECODE NUMERIC CHARACTER STRING  
C INTEGER LCHREQ 3 LOCATION OF CHARACTER EQUAL TO SEARCH CHARACTER

**GETS  
003**

**N-109**

DAN PACKAGE APPENDIX H  
UTILITY ROUTINES

GETS  
804

```

CALL TRACE( 'GETSAL')
CALL SREADS      & SPAN IF ALLOWED & REQUIRED
CALL GETOKH(KHTEMP.(1),LUSLLM, LUSLLM.LUSING)      & GET 1ST CHAR
IF(LUSHAX.OT.0) GO TO 220
CALL WARNS( MSHARN)
GO TO 900
220 IF(.NOT.(TRUAL(KHTEMP.1.1)))
& .OR.(TRUAL(LUSING,LUSLOC,LUSLEN))) GO TO 240
CALL WARNS( MSHARN)
GO TO 900
240 LUSLEN=0      & PREPARE TO GET SAME FIELD AGAIN
CALL GETOKH(KHFLO.(KHLEN),LUSLLM, LUSLLM.LUSING)
GO TO 900

```

C  
C  
C  
C  
C  
C

```

ENTRY GETSKH:  & GET CHARACTER STRING DATA FIELD FROM UNIT 5
0 KHFLD.      & RETURN AS CHARACTER STRING (UNCHANGED IF MISSING)
(KHLEN.      & LENGTH IN CHARS (WILL BE PADDED WITH BLANKS TO WORD 80))
-
1 MSHARN)      & WARNING MESSAGE IF MISSING (* MEANS REQUIRED)
-----

```

C  
C  
C

```

CALL TRACE( 'GETSKH')
CALL SREADS      & SPAN IF ALLOWED & REQUIRED
CALL GETOKH(KHFLO.(KHLEN),LUSLLM, LUSLLM.LUSING)
IF(LUSHAX.LE.0) CALL WARNS( MSHARN)
GO TO 900

```

C  
C  
C  
C  
C  
C

```

ENTRY GETSRL:  & GET REAL DATA FIELD FROM UNIT 5
0 RLFLD.      & RETURN AS REAL (UNCHANGED IF MISSING OR INVALID)
-
1 RLCOEF.      & COEFFICIENT BY WHICH INPUT FIELD IS TO BE MULTIPLIED
1 RLMIN,RLMAX. & MINIMUM/MAXIMUM VALID REAL
1 MSHARN)      & WARNING MESSAGE IF MISSING/INVALID (* MEANS REQUIRED)
-----

```

C  
C  
C

```

CALL TRACE( 'GETSRL')
KEYKOD='RL'
GO TO 900

```

C  
C  
C  
C  
C  
C

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

GETS  
000

```

      ENTRY GETSFR:  & GET FRACTION DATA FIELD FROM UNIT 5
      0 RLFLD.      & RETURN AS REAL (UNCHANGED IF MISSING OR INVALID)
      *
      1 RLCOEF.      & COEFFICIENT BY WHICH INPUT FIELD IS TO BE MULTIPLIED
      1 RLMIN,RLMAX. & MINIMUM/MAXIMUM VALID REAL
      1 MSHARN)      & WARNING MESSAGE IF MISSING/INVALID (* MEANS REQUIRED)
      -----
C
C
C
      CALL TRACE(  'GETSFR')
      KEYKOD='FR'
      GO TO 500
C
C
C
C
C
      ENTRY GETSSX:  & GET DECIMAL-CODED SEXAOGINARY DATA FIELD FROM UNIT 5
      0 RLFLD.      & RETURN AS REAL (UNCHANGED IF MISSING OR INVALID)
      *
      1 RLCOEF.      & COEFFICIENT BY WHICH INPUT FIELD IS TO BE MULTIPLIED
      1 RLMIN,RLMAX. & MINIMUM/MAXIMUM VALID REAL
      1 MSHARN)      & WARNING MESSAGE IF MISSING/INVALID (* MEANS REQUIRED)
      -----
C
C
C
      CALL TRACE(  'GETSSX')
      KEYKOD='SX'
C
C
C
C
      DECODE/SCALE/CHECK REAL/FRACTION/SEXAOGINARY
C
      500 CALL SREADS  & SPAN IF ALLOWED & REQUIRED
      CALL GETOKH(KHTEMP,(24),LUSLLM, LUSLLM,LUSIMO)
      IF(LUSMAX.GT.0) GO TO 520
      CALL WARNS(  MSHARN)
      GO TO 500
      520 IF(KEYKOD.EQ.'SX')
      & CALL PUTCHR(KHTEMP,(LCHREG(KHTEMP,1,24,'/')),  ':')
      IF(KEYKOD.EQ.'SX')
      & CALL PUTCHR(KHTEMP,(LCHREG(KHTEMP,1,24,'/')),  ':')
      CALL DCODE(INTMP, RLTEMP, KODTYP,
      * KHTEMP,(1),(LCHREG(KHTEMP,1,24,' ') - 1))
      RLTEMP=RLTEMP*RLCOEF
      IF((RLTEMP.LT.RLMIN).OR.(RLTEMP.GT.RLMAX)) KODTYP='ERR'
      IF((KODTYP.EQ.'IN').OR.
      & (KODTYP.EQ.'RI').OR.
      & (KODTYP.EQ.KEYKOD)) GO TO 540
      CALL WARNS(  MSHARN)
      GO TO 500
      540 RLFLD=RLTEMP
      GO TO 500
C
C

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

GETS  
000

C  
C  
C  
C

ENTRY GETSIN: 0 GET INTEGER DATA FIELD FROM UNIT 9  
0 INFLD. 0 RETURN AS INTEGER (UNCHANGED IF MISSING OR INVALID)  
1 INMIN,INMAX. 0 MINIMUM/MAXIMUM VALID INTEGER  
1 MSGARN) 0 WARNING MESSAGE IF MISSING/INVALID (0 MEANS REQUIRED)  
-----

C  
C  
C

CALL TRACE( 'GETSIN')  
CALL SREADS 0 SPAN IF ALLOWED & REQUIRED  
CALL GETOKN(KHTEMP,(24),LUSLLM, LUSLLM,LUSINO)  
IF(LUSHAX.GT.0) GO TO 820  
CALL WARNS( MSGARN)  
GO TO 900  
820 CALL DCODE(INTERP, RLTEMP, KODTYP,  
- KHTEMP,(1),(LCHREQ(KHTEMP,1,24,' ')-1))  
IF((INTERP.LT.INMIN).OR.(INTERP.GT.INMAX)) KODTYP='ERR'  
IF(KODTYP.EQ.'IN') GO TO 840  
CALL WARNS( MSGARN)  
GO TO 900  
840 INFLD=INTERP  
GO TO 900

C  
C  
C  
C  
C  
C

ENTRY UNGETS 0 BACK UP 1 FIELD ON UNIT 9  
-----

C  
C  
C

CALL TRACE( 'UNGETS')  
LUSLEN=0 0 PREPARE TO GET SAME FIELD AGAIN

C  
C  
C  
C

000 DONE

000 RETURN  
END

**HOWE T**  
**201**

```

1 LASTND.      0 LAST HEADING LINE TO WRITE (0/1/2)
1 NUNIT)      0 OUTPUT UNIT

```

\_\_\_\_\_

```

C
C      E M SCHLOSSER      LEC      07/17/73      ORIGINAL CODE
C      E M SCHLOSSER      LEC      02/07/80      UPGRADE DOCUMENTATION
C      E M SCHLOSSER      LEC      05/20/80      WRITE ASCII VT TO TERMINAL

```

**C PROJECT. THEN WRITE HEADING LINES 0 THRU LASTING ON UNIT.**

**C**      **FORMAT SPECIFICATIONS ASSUME 6 CHARACTERS PER INTEGER.**

```

C      EAPRNT      D WRITE ASCII STRING TO TERMINAL

```

**G NONE.**

INCLUDE KONXQT.LIST      3 COMMON PROGRAM EXECUTION SWITCHES, COUNTERS  
INCLUDE KONALT.LIST      3 COMMON ALTERNATE PRINT FILE COUNTERS, POINTERS

**INTEGER LSKIP      A (NUMBER+1) OF LINES TO SKIP AFTER PAGE EJECT**

2

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**NUNIT  
002**

**C PAGE EJECT AND PROVIDE TOP MARGIN**

```

C
      LSKIP=0
      IF((NUNIT.EQ.0).AND.(NBATCH.NE.1)) LSKIP=0
      IF((NUNIT.EQ.0).AND.(NBATCH.EQ.1)) LSKIP=1
      IF(LSKIP.EQ.0) CALL EAPRNT(1,1,1)      & ASCII JT TO TERMINAL
      IF(LSKIP.NE.0) WRITE(NUNIT,105)
105  FORMAT('1')      & ASCII FF TO ANYTHING ELSE
      DO 120 L=1,LSKIP
          WRITE(NUNIT,115)
115  FORMAT(' ')
120  CONTINUE
      IF(LASTND.OT.0) GO TO 300

```

**C**

**C**

**C PRINT SINGLE LINE HEADING**

**C**

```

      IF((NUNIT.EQ.0).AND.      & UNIT 0 (PRINTS)
& (NBATCH.NE.0))      & BATCH RUN (SPOOLED FOR PRINTER)
&      GO TO 400      & DON'T PRINT HEADING. BECAUSE SYMBIONT WILL
      WRITE(NUNIT,225) JRPIDT
225  FORMAT(10A6)
      GO TO 400

```

**C**

**C**

**C PRINT MULTI-LINE HEADING**

**C**

```

300  NPAGE=NPAGE+1
      NLINE=10
      WRITE(NUNIT,325) JRPIDT,NWNDOW,NIT,NPAGE
325  FORMAT(5X,10A6,2X,J3,'-',J1,'-',J4)
      WRITE(NUNIT,345) (JMDQ(N),N=1,12)
345  FORMAT(5X,12A6)
      IF(LASTND.EQ.1) GO TO 400
      WRITE(NUNIT,345) (JMDQ(N),N=13,24)

```

**C**

**C**

**C SKIP LINES AFTER HEADING**

**C**

```

400  DO 450 N=2,LASTND
          WRITE(NUNIT,425)
425  FORMAT(1X)
450  CONTINUE
      RETURN
      END

```

**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**184350  
001**

```
.      SUBROUTINE 184350(      & GET 4350-WORD 1-BANK ARRAY & CALL EXTERNAL SUBROUTINE
.      & NAMSUB.              & NAME OF EXTERNAL SUBROUTINE TO CALL
.      & DUMMY.              & DUMMY ARG (4350-WORD 1-BANK ARRAY PASSED AS 1ST ARG)
.      & ARG2.              & 2ND ARG ON CALL TO NAMED SUBROUTINE (OPTIONAL)
.      & ARG3.              & 3RD ARG ON CALL TO NAMED SUBROUTINE (OPTIONAL)
.      & . . .
.      & ARON)              & NTH ARG ON CALL TO NAMED SUBROUTINE (OPTIONAL)
.      -----
```

. (E M SCHLOSSER)

```
.      DIMENSION ARRAY(4350)      & PUTS ARRAY IN 1-BANK. UNLIKE UNIVAC FORTRAN V
.      CALL NAMSUB(ARRAY,ARG2,ARG3. . . ,ARON)
.      RETURN
```

. THIS SUBROUTINE IS USED WHEN A LARGE ARRAY MUST BE PUT IN THE 1-BANK  
. IN ORDER TO BALANCE THE STRUCTURE OF OVERLAY SEGMENTS IN A BANK-IMPLIED  
. COLLECTION.

S(01)	AXRS		
184350*	LXI	X11,ONE	. INCREMENT ARGUMENT POINTER BY 1 ON EXIT
	LA	A2,(ARRAY)	. ADDRESS OF 1-BANK ARRAY
	SA	A2,1,X11	. BECOMES ADDRESS OF 2ND ARG
	J	*0,*X11	. JUMP TO 1ST ARG & POINT TO 2ND ARG
ONE	*	1	
GAP1	RES	1	. NO MAN'S LAND
ARRAY	RES	4350	. ARRAY IS IN 1-BANK INSTEAD OF 0-BANK
GAP2	RES	1	. NO MAN'S LAND

. SAMPLE CALLING SEQUENCE

```
.      -----
.      EXTERNAL SUBA
.      CALL 184350(SUBA,DUMMY,X2,X3)
.      . . .
.      SUBROUTINE SUBA(ARRAY,X2,X3)
.      DIMENSION ARRAY(4350)
.      . . .
.      END
```



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

IDERT  
001

```

SUBROUTINE IDERT(  & PRINT SHORT ID FOR ERTS SCENE
1 (UNIT)          & OUTPUT UNIT
-----
C
C
C (E H SCHLOSSER)
C
C
C EXTERNAL SUBROUTINES/FUNCTIONS CALLED
C -----
C
C      SPLIT
C
C
C      INCLUDE KOMNER.LIST  & MUST HAVE BEEN PREVIOUSLY LOADED BY CALL TO OPEN3
C
C      CALL SPLIT(PITDEG,ISPITD,INPITD,APITD)
C      CALL SPLIT(ROLDEG,ISROLD,INROLD,AROLD)
C      WRITE(UNIT,125) NERTS,NCCT.
1      NERDAY,NERMON,NERYR.
2      NERSAM.
3      NERSEL.
4      ISPITD,INPITD,APITD.
5      ISROLD,INROLD,AROLD
125 FORMAT('  ',E-.11,J4,'-',J5,'-',J11,
1      ' ',J12,A3,J2,
2      ' ',J14,'SAMPLES',
3      ' SUNEL',J2,
4      ' PITCH',A1,J11,F3.2,
5      ' ROLL',A1,J11,F3.2)
      RETURN
      END

```

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

IDERTS  
001

```

SUBROUTINE IDERTS( 3 PRINT COMPLETE ID FOR ERTS SCENE
-
1 IUNIT) 3 OUTPUT UNIT
-----
C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      09/17/73      ALGORITHM CODING
C      J C CRISP          LEC      10/04/79      ADDITIONAL ERTS SCENE PARAMETERS
C      J C CRISP          LEMSCO   01/18/80      LOW/HIGH DEFINED LINES/SAMPLES
C
C
C METHOD
C -----
C
C      WRITE INFORMATION EXTRACTED FROM ID. HEADER. AND ANNOTATION RECORDS
C      FROM A TAPE.
C
C      ERTS CONVENTIONS FOR ATTITUDE AND HEADING ARE AS FOLLOWS:
C      POSITIVE PITCH IS NOSE DOWN
C      POSITIVE ROLL IS CLOCKWISE VIEWED FROM ABOVE
C      POSITIVE YAW IS COUNTERCLOCKWISE VIEWED FROM ABOVE
C      POSITIVE HEADING IS CLOCKWISE VIEWED FROM ABOVE
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE
C
C
C EXTERNAL REFERENCES
C -----
C
C      SPLIT      3 SPLIT REAL INTO SIGN. INTEGER. DECIMAL
C
C
C EXCEPTIONS
C -----
C
C      1. OPEN3 MUST HAVE BEEN CALLED BEFORE CALLING THIS ROUTINE.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST      3 COMMON PROGRAM EXECUTION SWITCHES.COUNTERS
C      INCLUDE KOMNER.LIST      3 ERTS SCENE PARAMETERS
C      INCLUDE KOMIHW.LIST      3 DEFINE INPUT WINDOW PACKET
C      INCLUDE MINDEF.LIST      3 DEFINE TWO-DIMENSIONAL WINDOW
C
C
C LOCAL DECLARATIONS
C -----

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

10ERTS  
002

```
C
      INTEGER MSALLO,MSALHI      8 LOW AND HIGH DEFINED LINES IN FULL SCENE
      INTEGER MSASLO,MSASHI      8 LOW AND HIGH DEFINED SAMPLES IN FULL SCENE
      INTEGER IOROH0             8 ORBIT HEADING
      INTEGER NCHAN              8 CHANNEL NUMBER
      INTEGER ITEMP (10)         8 TEMPORARY BUFFER OF CHANNEL NUMBERS
```

```
C
C
C PROCEDURE
C -----
C
```

```
      CALL TRACE
      IOROH0=MHYDE0
      CALL SPLIT (PITDE0,ISPITD,INPITD,APITD)
      CALL SPLIT (ROLDE0,ISROLD,INROLD,AROLD)
```

```
C
C
C COMPUTE RANGE OF LINES & SAMPLES IN FULL SCENE
C
```

```
      MSALLO=CTRLIN-0.5*(NERLIN-1)+0.5
      MSALHI=CTRLIN+0.5*(NERLIN-1)+0.5
      MSASLO=CTRSAM-0.5*(NERSAM-1)+0.5
      MSASHI=CTRSAM+0.5*(NERSAM-1)+0.5
```

```
C
C
C
      WRITE (IUNIT,200) NERSAT,NERSEN,
0      NERADN,NERPAT,NERROW,
1      NERTS,
2      NCCT,NCCTOT,
3      MSAIHW(WLIN,WMIN),MSAIHW(WLIN,WMAX),
3      MSALLO,MSALHI,NERLIN,
4      MSAIHW(MSAM,WMIN),MSAIHW(MSAM,WMAX),
4      MSASLO,MSASHI,NERSAM,
5      NERDAY,NERMON,NERYR
```

```
C
200 FORMAT ('0'.3A6/
0      'OPATH-ROW:   '.A1.1X.A3.1X.A3/
1      ' SCENE:     '.11.J4.'-''.J5/
2      ' STRIP:     '.6X.11.' OF '.11/
3      ' LINES:     '.14.' TO '.14.' OF '.14.' TO '.14.
3      '              3X.'(SIZE=''.14.'')'/
4      ' SAMPLES:   '.14.' TO '.14.' OF '.14.' TO '.14.
4      '              3X.'(SIZE=''.14.'')'/
5      ' DATE:      '.14.1X.A3.1X.J2)
```

```
C
C
      WRITE (IUNIT,300) CTRLAT,CTRLON,
0      CTRLIN,CTRSAM,
1      OIRLAT,OIRLON,
2      NERSEL,
3      NERSAZ,
4      IOROH0,
5      ALTKH,
6      ISPITD,INPITD,APITD,
7      ISROLD,INROLD,AROLD
```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

IDENTS  
003

```

C
300 FORMAT (' CENTER(DEG): ',F9.4,2X,F9.4/
0          ' CENTER(SCAN): ',F8.1,5X,F8.1/
1          ' NAQIR: ',F9.4,2X,F9.4/
2          ' SUN EL: ',.14/
3          ' SUN AZ: ',.14/
4          ' HDQ-YAH: ',.14/
5          ' ALT: ',F5/
6          ' PITCH: ',.A1,11,F3.2/
7          ' ROLL: ',.A1,11,F3.2)

C
C
C LOAD TEMPORARY BUFFER WITH CHANNEL NUMBERS
C
DO 400 NCHAN=1,NERCHA
ITEMP(NCHAN)=NCHAN
400 CONTINUE

C
C
WRITE (IUNIT,500) NERCEO,
0          NERRES,
1          NERCOR

C
C
WRITE (IUNIT,550) (ITEMP(NCHAN), NCHAN=1,NERCHA)

C
C
WRITE (IUNIT,600) (NERBAN(NCHAN), NCHAN=1,NERCHA)

C
C
WRITE (IUNIT,650) (NERGAI(NCHAN), NCHAN=1,NERCHA)

C
C
WRITE (IUNIT,700) (NERTHO(NCHAN), NCHAN=1,NERCHA)

C
C
500 FORMAT (' GEOMETRY: ',.A3/
0          ' RESAMPLING: ',.A3/
1          ' CORRECTION: ',.A3)

C
C
550 FORMAT (' CHANNEL: ',.10(13))

C
C
600 FORMAT (' BAND: ',.10(A3))

C
C
650 FORMAT (' GAIN: ',.10(A3))

C
C
700 FORMAT (' MODE: ',.10(A3))

C
C
C NORMAL RETURN
C
RETURN

```

**OAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**IDERTS  
004**

**END**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

10LU3  
001

```

SUBROUTINE 10LU3( 3 PRINT SHORT ID FOR LOGICAL UNIT 3
.
1 (UNIT) 3 OUTPUT UNIT
-----
C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER   LEC  10/26/79   REQUIREMENTS
C      C A HELMKE     LEC  10/29/79   ALGORITHM DESIGN
C      C A HELMKE     LEC  10/29/79   ALGORITHM CODING
C
C
C METHOD
C -----
C
C      USE THE CHARACTER BUFFER ROUTINES TO FORMAT AN OUTPUT DISPLAY
C      FOR LOGICAL UNIT 3 INFORMATION.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      THE INTERNAL FILE NAME FOR AN ALTERNATE PRINT FILE REQUIRES 12
C      CHARACTERS WHICH IS 2 WORDS ON THE UNIVAC 1100.
C
C
C EXTERNAL REFERENCES
C -----
C
C      ERPRNT      3 PRINT STRING OF FIELDATA CHARACTERS
C      ERPRTA      3 WRITE STRING OF FIELDATA CHARACTERS TO ALT PRT FILE
C      CBS4IN      3 ENCODE INTEGER TO A VARIABLE-LENGTH STRING
C      CST4IN      3 CHARACTER STRING FOR INTEGER
C      CBINIT      3 INITIALIZE CHARACTER BUFFER
C      CB4CST      3 APPEND CHARACTERS TO CHAR BUFFER FROM CHAR STRING
C      CB4IN       3 ENCODE INTEGER & APPEND TO CHARACTER BUFFER
C      MOFATL      3 PRINT/LOG/TALLY 'FATAL ERROR' MESSAGE
C
C      DOUBLE PRECISION CBS4IN
C
C
C EXCEPTIONS
C -----
C
C      1.  IF IN DATA/CHECKOUT MODE AND OUTPUT UNIT IS NOT EQUAL 3, RETURN IS
C          IMMEDIATE WITH NO OUTPUT SINCE ALTERNATE PRINT FILES ARE NOT
C          AVAILABLE IN DATA/CHECKOUT MODE.
C
C      2.  ALTERNATE PRINT FILE OUT OF RANGE INDICATES A PROGRAMMER FATAL ERR.
C
C
C GLOBAL DECLARATIONS
C -----
C

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

IDLU3  
002

```

      INCLUDE KOMXQT.LIST      8 COMMON PROGRAM EXECUTION SWITCHES. COUNTERS
      INCLUDE KOHLU3.LIST      8 I/O AND UNPACKING DATA FOR MSS/RBV DATA
      INCLUDE ICBUF1.LIST      8 STANDARD CHARACTER BUFFER FOR CHARACTER
C                                8 BUFFER ROUTINES
      INCLUDE FIDF.LIST        8 MNEMONICS FOR POINTERS TO LOCATIONS IN
C                                8 IDFILE BUFFER
C
C
C LOCAL DECLARATIONS
C -----
C
      INTEGER IALTUN(2)        8 ENCODED 12 CHAR INTERNAL FILE NAME FOR ALT
C                                8 PRINT FILE
C
C
C
C PROCEDURE
C -----
C
      CALL TRACE
C
C
C IF WRITING TO ALTERNATE PRINT FILE THEN CHECK FILE & ENCODE UNIT NUMBER
C
      IF (IUNIT.EQ.8) GO TO 220
      IF (MDATAAC.NE.0) GO TO 900      8 DATA/CHECKOUT MODE-NO ALT PRT FILES
      IF (IUNIT.GE.10 .AND. IUNIT.LE.(10+MALTH-1)) GO TO 200
      CALL MDATAI(
-          'IDLU3 CALLED WITH INVALID PRINT FILE UNIT -'.
&          CS$IN(IUNIT) )
      GO TO 900
200      CALL CST4IN (IALTUN,1,12,IUNIT,1)
220 CONTINUE
C
C
C ENCODE OUTPUT LINE
C
      CALL C$INIT(ICBUF1)
      IF (LU3FID(FIDEQT).EQ.'TAPE'.OR.LU3FID(FIDEQT).EQ.'DISK') GO TO 240
      CALL CB4CST(ICBUF1,'NO EQUIPMENT',(1),12)
      GO TO 280
240 CALL CB4CST(ICBUF1,LU3FID(FIDEQT),(1),4)
      CALL CB4CST(ICBUF1,'',(1),2)
      CALL CB4CST(ICBUF1,LU3FID(FIDEQC),(1),4)
      CALL CB4CST(ICBUF1,'',(1),2)
      IF (LU3FID(FIDEQT).NE.'TAPE') GO TO 260
      CALL CB4CST(ICBUF1,LU3FID(FIDTRK),(1),1)
      CALL CB4CST(ICBUF1,'TRK',(1),5)
      CALL CB4CST(ICBUF1,LU3FID(FIDFPI),(1),4)
      CALL CB4CST(ICBUF1,'FPI REEL',(1),10)
      DO 255 IVOL = 1,LU3VHI
          IF (IVOL.NE.1) CALL CB4CST(ICBUF1,'',(1),1)
          CALL CB4CST(ICBUF1,LU3REL(IVOL),(1),6)
          IF (IVOL.EQ.LU3VOL) CALL CB4CST(ICBUF1,'',(1),1)
255      CONTINUE
      CALL CB4CST(ICBUF1,'',(1),1)

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

1DLU3  
003

```
200 CALL CB4CST(1CBUF1,LU3FID(FIDBFN),(1),3)
    CALL CB4CST(1CBUF1,' BUF FMT',(1),0)
C
C
C PRINT LINE
C
200 IF (1UNIT.EQ.0) CALL ERPRNT(1,20,1CBUF1)
    IF (1UNIT.NE.0) CALL ERPRTA(1ALTUN,1,20,1CBUF1)
C
C
200 RETURN
END
```



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

IDUP  
001

. INTEGER FUNCTION  
. 0 8 INTEGER DUPLICATE (INDIRECT REFERENCE TO OPTIONAL ARGUMENT)  
. = IDUP(  
. 1 INTEGER) 8 INTEGER  
. -----

. HISTORY  
. -----

. E M SCHLOSSER LEC 09/27/79 ORIGINAL CODE

. METHOD  
. -----

. LOAD ARGUMENT INTO REGISTER A0.

. MACHINE-DEPENDENT CODE  
. -----

. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS.  
. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
. BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT  
. COMPILERS (EG., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

. EXTERNAL REFERENCES  
. -----

. NONE

. EXCEPTIONS  
. -----

. NONE

. GLOBAL DECLARATIONS  
. -----

. NONE.

. AXRS . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES  
S(00) . 1-BANK  
IDUP\* LA A0.\*0.X11 .  
J 2.X11 .  
END

**DAM PACKAGE APPENDIX M  
UTILITY ROUTINES**

**INVR1  
001**

**SUBROUTINE INVR1( 8 ADD ORIGIN TO ENVELOPE (INTEGER ONLY)  
U NWWPKT) 8 WINDOW PACKET (INTEGER)**

**-----**

**C  
C  
C (E H SCHLOSSER)  
C  
C**

**DIMENSION NWWPKT(2,1)  
INCLUDE WINDEF.LIST**

**C  
C**

**NWWPKT(1,WMIN)=NWWPKT(1,WMIN)+NWWPKT(1,WORIG)  
NWWPKT(1,WMAX)=NWWPKT(1,WMAX)+NWWPKT(1,WORIG)  
NWWPKT(2,WMIN)=NWWPKT(2,WMIN)+NWWPKT(2,WORIG)  
NWWPKT(2,WMAX)=NWWPKT(2,WMAX)+NWWPKT(2,WORIG)  
RETURN  
END**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

INVMIN  
001

```

      SUBROUTINE INVMIN  S COMPUTE ENVELOPE FOR INTEGER WINDOW
      U NWWPKT)          S WINDOW PACKET (INTEGER)
      -----
C
C (E M SCHLOSSER)
C
C
C EXTERNAL SUBROUTINES/FUNCTIONS CALLED
C -----
C
C      MDEFATL
C
C
C      DIMENSION NWWPKT(2,1)
C      INCLUDE WINDEF.LIST
C
C INITIALIZE ENVELOPE
C
      NWWPKT(1,WMIN)=+(2**32)
      NWWPKT(1,WMAX)=--(2**32)
      NWWPKT(2,WMIN)=+(2**32)
      NWWPKT(2,WMAX)=--(2**32)
C
C
C GET POINTERS TO FIRST AND LAST VERTICES
C
      MINVER=MVER+1
      MAXVER=NWWPKT(MUSED,WHEAD)
      IF(MAXVER.GT.MINVER) GO TO 200
      IF(MAXVER.NE.MINVER) CALL MDEFATL('BAD ENTRY NODE POINTER')
      MAXVER=MINVER+1
      NWWPKT(1,MAXVER)=0      S THE OTHER VERTEX IS THE ORIGIN
      NWWPKT(2,MAXVER)=0
C
C
C FIND THE ENVELOPE
C
      200 DO 300 NVER=MINVER,MAXVER
      NWWPKT(1,WMIN)=MIN0(NWWPKT(1,WMIN),NWWPKT(1,NVER))
      NWWPKT(1,WMAX)=MAX0(NWWPKT(1,WMAX),NWWPKT(1,NVER))
      NWWPKT(2,WMIN)=MIN0(NWWPKT(2,WMIN),NWWPKT(2,NVER))
      NWWPKT(2,WMAX)=MAX0(NWWPKT(2,WMAX),NWWPKT(2,NVER))
      300 CONTINUE
      RETURN
      END

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

14KOLR  
001

```

SUBROUTINE 14KOLR( 8 INTEGER-COLOR-EQUIVALENT FOR COLOR
0 IKE.      8 INTEGER-COLOR-EQUIVALENT
.
1 KOLOR)    8 COLOR (12 CHARACTERS)
-----
C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      05/26/79      RQNTS/DESIGN/CODE
C
C
C
C METHOD
C -----
C
C      ASSIGN THE INTEGER-COLOR-EQUIVALENTS 0, 4, 8 TO THE CIE(1931)
C      TRISTIMULUS PRIMARY ADDITIVE COLORS BLUE, GREEN, RED, RESPECTIVELY.
C      ASSIGN INTERMEDIATE INTEGER-COLOR-EQUIVALENTS TO INTERMEDIATE COLORS.
C      (NOTE THAT BLACK (NO INFORMATION) AND WHITE (NO DATA) ARE NOT INCLUDED.)
C      IMPLEMENTED BY EXHAUSTIVE SEARCH ON SIGNIFICANT CHARACTERS OF KOLOR:
C          FIRST CHARACTER
C          DASH (-), IF PRESENT
C          CHARACTER FOLLOWING DASH, IF PRESENT
C
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C
C EXTERNAL REFERENCES
C -----
C
C      MOVCSY      8 MOVE CHARACTER STRING
C      INTEGER LCHREQ 8 LOCATION IN STRING OF CHARACTER EQUAL
C      INTEGER LENPAD 8 LENGTH IN CHARS. PADDED TO NEXT WORD BOUNDARY
C
C
C
C EXCEPTIONS
C -----
C
C      1. IKE IS UNCHANGED IF KOLOR IS NOT FOUND.
C
C
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER KL      8 COLOR (SIGNIFICANT CHARACTERS)

```

**DAN PACKAGE APPENDIX N.  
UTILITY ROUTINES**

**14KOLR  
002**

```

      INTEGER LDASH      % LOCATION OF FIRST DASH (-) IN KOLOR

C
C
C PROCEDURE
C -----
C
C PUT SIGNIFICANT CHARACTERS FROM KOLOR INTO KL
C
      CALL MOVCST(KL.(1).(LENPAD(3)).
      KOLOR.(1).(3).% ' ' % GET 1ST 3 CHARS
      LDASH=LCHREG(KOLOR.(1).(12).%-) % LOCATION OF DASH
      IF(LDASH.NE.0) CALL MOVCST(KL.(2).(LENPAD(3)).1).
      KOLOR.(LDASH).(2).% ' ' % INSERT DASH & NEXT CHR

C
C
C ASSIGN CORRESPONDING I-K-E
C
      IF((KL.EQ.'B ').OR.
      & (KL.EQ.'BL ').OR.
      & (KL.EQ.'BLU')) IKE=0      % BLUE      % PRIMARY ADDITIVE COLOR
      IF((KL.EQ.'C-B').IKE=1      % CYAN-BLUE
      IF((KL.EQ.'C ').OR.
      & (KL.EQ.'CY ').OR.
      & (KL.EQ.'CYA')) IKE=2      % CYAN
      IF((KL.EQ.'A ').OR.
      & (KL.EQ.'AQ ').OR.
      & (KL.EQ.'AQU')) IKE=3      % AQUA
      IF((KL.EQ.'O ').OR.
      & (KL.EQ.'OR ').OR.
      & (KL.EQ.'ORE')) IKE=4      % GREEN      % PRIMARY ADDITIVE COLOR
      IF((KL.EQ.'Y-O')) IKE=5      % YELLOW-GREEN
      IF((KL.EQ.'Y ').OR.
      & (KL.EQ.'YE ').OR.
      & (KL.EQ.'YEL')) IKE=6      % YELLOW
      IF((KL.EQ.'O ').OR.
      & (KL.EQ.'OR ').OR.
      & (KL.EQ.'ORA')) IKE=7      % ORANGE
      IF((KL.EQ.'R ').OR.
      & (KL.EQ.'RE ').OR.
      & (KL.EQ.'RED')) IKE=8      % RED      % PRIMARY ADDITIVE COLOR
      IF((KL.EQ.'R-M')) IKE=9      % RED-MAGENTA
      IF((KL.EQ.'M ').OR.
      & (KL.EQ.'MA ').OR.
      & (KL.EQ.'MAG')) IKE=10      % MAGENTA

C
C
C COMMON RETURN
C
      999 RETURN
      END

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

JACHX/MATHPACK  
001

SUBROUTINE JACHX( 3 JACOBI ITERATION TO FIND EIGENVALUES & EIGENVECTORS  
U SYMEVL. 3 1: SYMMETRIC MATRIX 0: EIGENVALS ON DIAGONAL  
O EIGVEC. 3 EIGENVECTORS  
I NRCDIM. 3 NUMBERS OF ROWS & COLUMNS DIMENSIONED  
I NRCUSE. 3 NUMBER OF ROWS & COLUMNS USED  
I CONVRO. 3 CONVERGENCE TOLERANCE  
U NITER. 3 1: MAXIMUM ITERATIONS 0: ACTUAL NUMBER OF ITERATIONS  
I S. 3 RETURN TRANSFER LABEL IF NO CONVERGENCE  
I KFLAG) 3 1 = FIRST CALL, 2 = SUBSEQUENT CALL

C  
C  
C

.....  
\*\*\*\*\* UNIVAC 1100 MATHPACK ROUTINE \*\*\*\*\*  
\*\*\*\*\* SOURCE CODE IS NOT AVAILABLE \*\*\*\*\*  
.....

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

JOIN2N  
001

```

SUBROUTINE JOIN2N:  JOIN ADJACENT DETECTION RECORDS IN SAME LINE
O MPXBUF.  JOIN DETECTION BUFFER IN PBXDEF FORMAT
O JSTAT.  JOIN STATUS CODES
C          ' ' NORMAL COMPLETION
C          'ERR' ERROR COMPLETION
C
C
C      I MPXLFT.  LEFT PORTION OF DETECTION BUFFER (IDENTICAL TO MPXBUF)
C              IN PBXDEF FORMAT
C      I MPXRHT)  RIGHT PORTION OF DETECTION BUFFER IN PBXDEF FORMAT
C      -----
C
C
C HISTORY
C -----
C
C      MARY TOMPKINS      LEC      12/08/79      REQUIREMENTS
C      MARY TOMPKINS      LEC      12/19/79      ALGORITHM DESIGN
C      MARY TOMPKINS      LEC      12/19/79      ALGORITHM CODING
C
C
C METHOD
C -----
C
C      IF MPXLFT(PXBINT)<>MPXRHT(PXBINT) ISSUE M0FATL AND RETURN.
C      IF MPXBUF<>MPXBUF ISSUE M0FATL AND RETURN.
C      KLSTYP = 'DEN'
C          ACCORDING TO (PXLJ01), (PXHJ01) IN PREAMBLES OF MPXLFT AND
C          MPXRHT PERFORM SEAMING OF THE TWO BUFFERS. SET
C          MPXBUF(PXHJ01) = MPXRHT(PXHJ01).
C      FOR ALL KLSTYP
C          ADJUST MPXBUF(PXHSA1) AND MPXBUF(PXHBIN) TO REFLECT THE
C          HIGHEST SAMPLE AVAILABLE AND BIN CONTAINING THE HIGHEST
C          AVAILABLE SAMPLE. SET (PXQUAL) = MAX0(PXQUAL) OF THE INPUT
C          BUFFERS. SHIFT STARTING AT MPXRHT(PXBINS) NUMBIN OF DATA
C          TO MPXBUF BUFFER BEGINING AT MPXBUF(PXHBIN)+1 POSITION.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      GETBYT      JOIN GET NON-NEG INTEGER FROM BYTE IN BYTE STRING
C      GETICE      JOIN GET INT CHAR EQUIVALENT FROM CHAR STRING
C      GETINT      JOIN GET INTEGER FROM INTEGER STRING
C      PUTBYT      JOIN PUT NON-NEG INTEGER INTO BYTE
C      PUTICE      JOIN PUT INT CHAR EQUIVALENT INTO CHAR STRING
C      PUTINT      JOIN PUT INTEGER INTO INTEGER STRING
C      MOVBSY      JOIN MOVE BYTE STRING
C      MOVCSY      JOIN MOVE CHARACTER STRING
C      MOVIST      JOIN MOVE INTEGER STRING
C

```

DAN PACKAGE APPENDIX H  
UTILITY ROUTINES

JOIN2N  
002

```

C
C EXCEPTIONS
C -----
C
C     THE FOLLOWING CONDITIONS GENERATE THE DIAGNOSTIC SHOWN:
C           CONDITION                                DIAGNOSTIC
C     MPXLFT(PXBINT)<>MPXRHT(PXBINT)                MDFATL
C     MPXBUF<>MPXLFT                                MDFATL
C
C
C GLOBAL DECLARATIONS
C -----
C
C     INCLUDE KOMXQT.LIST      3 COMMON PROGRAM EXECUTION SWITCHES.COUNTERS
C     INCLUDE PXBDEF.LIST     3 BUFFER STRUCTURE DEFINITION
C     INCLUDE KOMKLS.LIST     3 COMMON CLASSIFICATION INFO
C
C
C LOCAL DECLARATIONS
C -----
C
C     INTEGER MPXBUF(1)      3 ARGUMENT
C     INTEGER MPXLFT(1)     3 ARGUMENT
C     INTEGER MPXRHT(1)     3 ARGUMENT
C     INTEGER NUMBIN        3 NUMBER OF BINS TO SHIFT
C     INTEGER JLOSAM        3 JOIN LOW JOIN TO LEFT SAMPLE
C     INTEGER JHISAM        3 JOIN HI JOIN TO RIGHT SAMPLE
C     INTEGER NXTBIN        3 NEXT BIN AFTER MPXLFT
C     INTEGER IZERO/0/      3 BINARY ZERO
C
C
C PROCEDURE
C -----
C
C
C SET STATUS CODE TO DEFAULT VALUE AND CHECK FOR
C EQUALITY BETWEEN BUFFERS
C
C     JSTAT = 'ERR'
C
C     IF(LOC(MPXLEFT).NE.LOC(MPXBUF)) CALL MDFATL(
C       = 'MPXLFT<>MPXBUF -- JOIN2N')
C     IF(LOC(MPXLEFT).NE.LOC(MPXBUF)) GO TO 900
C
C     IF(MPXLEFT(PXBINT).NE.MPXRHT(PXBINT)) CALL MDFATL(
C       = 'DIFFERENCE IN BIN TYPES -- JOIN2N')
C     IF(MPXLEFT(PXBINT).NE.MPXRHT(PXBINT)) GO TO 900
C
C
C RESET STATUS. TEST KLSYTP
C
C     JSTAT = . . .
C     IF(KLSYTP.NE.'DEN') GO TO 300
C
C

```



DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

JOIN2N  
003

C PREFORM SEARCHING OF BUFFERS

C  
IF(MPXLFT(PXBINT).EQ.'BYT') CALL GETBYT(JLOSAM,  
= MPXLFT(PXBINS),MPXLFT(PXHBIN))

C  
IF(MPXLFT(PXBINT).EQ.'CHR') CALL GETICE(JLOSAM,  
= MPXLFT(PXBINS),MPXLFT(PXHBIN))

C  
IF(MPXLFT(PXBINT).EQ.'INT') CALL GETINT(JLOSAM,  
= MPXLFT(PXBINS),MPXLFT(PXHBIN))

C  
JLOSAM = JLOSAM + MPXRHT(PXLJOI)

C  
IF(MPXLFT(PXBINT).EQ.'BYT') CALL PUTBYT(  
= MPXLFT(PXBINS),MPXLFT(PXHBIN), JLOSAM)

C  
IF(MPXLFT(PXBINT).EQ.'CHR') CALL PUTICE(  
= MPXLFT(PXBINS),MPXLFT(PXHBIN), JLOSAM)

C  
IF(MPXLFT(PXBINT).EQ.'INT') CALL PUTINT(  
= MPXLFT(PXBINS),MPXLFT(PXHBIN), JLOSAM)

C  
IF(MPXRHT(PXBINT).EQ.'BYT') CALL GETBYT(JHISAM,  
= MPXRHT(PXBINS),MPXRHT(PXLBIN))

C  
IF(MPXRHT(PXBINT).EQ.'CHR') CALL GETICE(JHISAM,  
= MPXRHT(PXBINS),MPXRHT(PXLBIN))

C  
IF(MPXRHT(PXBINT).EQ.'INT') CALL GETINT(JHISAM,  
= MPXRHT(PXBINS),MPXRHT(PXLBIN))

C  
JHISAM = JHISAM + MPXLFT(PXHJOI)

C  
IF(MPXRHT(PXBINT).EQ.'BYT') CALL PUTBYT(  
= MPXRHT(PXBINS),MPXRHT(PXLBIN), JHISAM)

C  
IF(MPXRHT(PXBINT).EQ.'CHR') CALL PUTICE(  
= MPXRHT(PXBINS),MPXRHT(PXLBIN), JHISAM)

C  
IF(MPXRHT(PXBINT).EQ.'INT') CALL PUTINT(  
= MPXRHT(PXBINS),MPXRHT(PXLBIN), JHISAM)

C  
C  
C ADJUST MPXBUF PREAMBLE

C  
MPXBUF(PXHJOI) = MPXRHT(PXHJOI)  
300 MPXBUF(PXHSAM) = MPXRHT(PXHSAM)  
MPXBUF(PXQUAL) = MAX0(MPXLFT(PXQUAL),MPXRHT(PXQUAL))  
NXTBIN = MPXLFT(PXHBIN) + 1  
MPXBUF(PXHBIN) = MPXBUF(PXHSAM)-MPXBUF(PXLSAM)+MPXBUF(PXLBIN)

C  
C  
C CONCATENATE BUFFERS

C  
NUMBIN = MPXRHT(PXHBIN) - MPXRHT(PXLBIN) + 1

C

OAM PACKAGE APPENDIX N  
UTILITY ROUTINES

JOHN2N  
004

```
      IF(MPXBUF(PXBINT).EQ.'BYT') CALL MOVBST(
& MPXBUF(PXBINS).NXTBIN.NUMBIN.
- MPXRHT(PXBINS).MPXRHT(PXLBIN).NUMBIN.1ZERO)
C
      IF(MPXBUF(PXBINT).EQ.'CHR') CALL MOVCS(
& MPXBUF(PXBINS).NXTBIN.NUMBIN.
- MPXRHT(PXBINS).MPXRHT(PXLBIN).NUMBIN.' ')
C
      IF(MPXBUF(PXBINT).EQ.'INT') CALL MOVIST(
& MPXBUF(PXBINS).NXTBIN.NUMBIN.
- MPXRHT(PXBINS).MPXRHT(PXLBIN).NUMBIN.1ZERO)
C
900 RETURN
END
C
```

**KOLR41**  
**001**

```

C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      05/26/79      RQMTS/DESIGN/CODE
C
C
C METHOD
C -----
C
C      ASSIGN THE CIE(1931) TRISTIMULUS PRIMARY ADDITIVE COLORS BLUE, GREEN,
C      RED TO THE INTEGER-COLOR-EQUIVALENTS 0, 4, 8, RESPECTIVELY.
C      ASSIGN INTERMEDIATE COLORS TO INTERMEDIATE INTEGER-COLOR-EQUIVALENTS.
C      (NOTE THAT BLACK (NO INFORMATION) AND WHITE (NO DATA) ARE NOT INCLUDED.)
C      IMPLEMENTED BY TABLE LOOKUP.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      DIMENSION SPECIFICATION ASSUMES 6 CHARACTERS PER INTEGER.
C
C
C EXTERNAL REFERENCES
C -----
C
C      MOVCS1      8 MOVE CHARACTER STRING
C      INTEGER LENPAD      8 LENGTH IN CHARS. PADDED TO NEXT WORD BOUNDARY
C
C
C EXCEPTIONS
C -----
C
C      1. KOLOR IS UNCHANGED IF I(K) IS LESS THAN 0 OR GREATER THAN 10.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER KOLTBL(22)/      8 COLOR TABLE IN ORDER OF I-K-E'S
C      0 'BLUE      ' 'CYAN-BLUE      '
C      2 'CYAN      ' 'AQUA      '
C      4 'GREEN      ' 'YELLOW-GREEN'
C      8 'YELLOW      ' 'ORANGE      '

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

KOLR41  
002

```
      S 'RED          :. 'RED-MAGENTA '.  
      O 'MAGENTA     :/  
C  
C  
C PROCEDURE  
C -----  
C  
      IF((IKE.OE.0).AND.(IKE.LE.10))  
      &      CALL MOVCST(KOLOR.(1).(LENPAD(12)).  
      *      KOLTBL.(IKE+12+1).(12).* ')  
      RETURN  
      END
```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

KSPRED  
001

```

SUBROUTINE KSPRED:  & SPREAD COUNT FLAGS INTO INTERIOR UNDEFINED PIXELS
U IPIX.            & 1 PIXEL/WORD:
C                  M1 = MEAN VALUE (SIGNED)
C                  M2 = COUNT OF ELEMENTS DEFINING MEAN (UNSIGNED)
1 NCOL.            & NUMBER OF COLUMNS IN DIGITAL PICTURE
1 NROW.            & NUMBER OF ROWS IN DIGITAL PICTURE
C                  !!!LAST 2 ROWS ARE SCRATCH SPACE FOR ROUTINE!!!
1 KSPRED.          & DISTANCE TO SPREAD COUNT FLAGS FROM DEFINED PIXELS
1 KSHRNK.          & DISTANCE TO SPREAD COUNT FLAGS FROM REMOTE PIXELS
C -----
C (E H SCHLOSSER)
C
C
C
C DIMENSION IPIX(NCOL,NROW)  & LAST 2 ROWS ARE SCRATCH SPACE -- NOT PICTURE
C INCLUDE ASHDEF.LIST
C
C DEFINE KPX(1,J)=ASHM2(IPIX(1,J))  & COUNT OF ELEMENTS DEFINING MEAN
C
C DEFINE MPXPUT(1,J)=ASHM1(IPIX(1,J))  & ASSIGN MEAN VALUE (SIGN EXTENDED)
C DEFINE MPXGET(1,J)=IPIX(1,J)/2**18  & RETRIEVE MEAN VALUE (SIGN EXTENDED)
C CALL TRACE
C
C
C
C
C PHASE 1 -- FLAG ALL UNDEFINED PIXELS AS REMOTE
C
C
C IF(KSPRED.LE.0) GO TO 900
C NRMAX=NROW-2  & LAST 2 ROWS ARE SCRATCH SPACE -- NOT PICTURE!!!
C DO 140 NR=1,NRMAX
C DO 120 NC=1,NCOL
C KPX(NC,NR)=MIN0(KPX(NC,NR),2**17-1)  & MUST FORCE BIT 17 ZERO
C IF(KPX(NC,NR).EQ.0) KPX(NC,NR)=2**18-1  & TEMP REMOTE FLAG
120 CONTINUE
140 CONTINUE
C
C
C
C
C PHASE 2 -- SPREAD FLAGS FROM NON-REMOTE PIXELS TO REMOTE PIXELS
C
C
C SPREAD FLAGS BY KSPRED
C
C KPXFLO=2**17+1
C KPXFHI=KPXFLO+KSPRED-1
C DO 390 KPXFLO=KPXFLO,KPXFHI
C
C
C
C SCAN PICTURE
C

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

KSPRED  
002

```

      DO 300 NR=1,NRMAX
      DO 370 NC=1,NCOL
      IF(KPX(NC,NR).NE.2*(10-1)) GO TO 370      8 PIXEL IS DEFINED
C
C
C INSURE THAT NEIGHBORS ARE INSIDE PICTURE
C
      NRLO=MAX0(NR-1,1)
      NRHI=MIN0(NR+1,NRMAX)
      NCLO=MAX0(NC-1,1)
      NCHI=MIN0(NC+1,NCOL)
C
C
C SEARCH NEIGHBORS FOR NON-REMOTE PIXELS
C
      DO 340 NRX=NRLO,NRHI
      DO 330 NCX=NCLO,NCHI
      IF(KPX(NCX,NRX).GE.KPXFL0) GO TO 330
      KPX(NC,NR)=KPXFL0
      GO TO 370
      330 CONTINUE
      340 CONTINUE
C
C
C KEEP SCANNING
C
      370 CONTINUE
      380 CONTINUE
C
C
C KEEP SPREADING
C
      390 CONTINUE
C
C
C
C
C
C PHASE 3 -- SPREAD FLAOS FROM REMOTE PIXELS TO EXPOSED NON-REMOTE PIXELS
C
C
C SPREAD FLAOS BY KSHRNK
C
      IF(KSHRNK.EQ.0) GO TO 700
      KPXFLO=KPXFHI-KSHRNK+1
      DO 590 KPXFLO=KPXFHI,KPXFLO,-1
C
C
C SCAN PICTURE
C
      DO 580 NR=1,NRMAX
      DO 570 NC=1,NCOL
      IF(KPX(NC,NR).LT.KPXFLO) GO TO 570      8 PIXEL IS NOT YET EXPOSED
      IF(KPX(NC,NR).GT.2*(17+120)) GO TO 570  8 PIXEL IS REMOTE
C
C

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**KSPRED  
003**

```

C CHECK IF PIXEL IS ON EDGE OF PICTURE
C
    IF(NR.EQ.1) GO TO 560
    IF(NR.EQ.NRMAX) GO TO 560
    IF(NC.EQ.1) GO TO 560
    IF(NC.EQ.NCOL) GO TO 560
C
C
C SEARCH NEIGHBORS FOR REMOTE PIXELS
C
    DO 540 NRD=-1,+1
    DO 530 NCD=-1,+1
    IF(KPX(NC+NCD,NR+NRD).GT.KPXFLG+128) GO TO 560
530 CONTINUE
540 CONTINUE
    GO TO 570
C
C
C FLAG EXPOSED UNDEFINED PIXEL
C
560 KPX(NC,NR)=KPXFLG+128
C
C
C KEEP SCANNING
C
570 CONTINUE
580 CONTINUE
C
C
C KEEP SPREADING
C
590 CONTINUE
C
C
C
C PHASE 4 -- FLAG ALL REMAINING REMOTE PIXELS AS UNDEFINED
C
C
C SCAN PICTURE
C
700 DO 760 NR=1,NRMAX
    DO 770 NC=1,NCOL
    IF(KPX(NC,NR).GT.2**17+128) KPX(NC,NR)=0
770 CONTINUE
760 CONTINUE
C
C
C
C
C END OF ROUTINE
C
C
900 RETURN
    END

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

LOOK41  
001

```
.      SUBROUTINE LOOK41( I LINE OF BOX DIGIT FOR INTEGER
.      0 LBOXD.      8 12-CHARACTER LINE OF 12-LINE BOX DIGIT
.      "
.      I IDIGIT.      8 1-DIGIT INTEGER
.      I KHRFOR.      8 FOREGROUND CHARACTER FOR BOX DIGIT
.      I KHRBAK.      8 BACKGROUND CHARACTER FOR BOX DIGIT
.      I LINE)      8 LINE NUMBER (1 THRU 12)
.      -----
```

HISTORY  
-----

```
.      E M SCHLOSSER      LEC      31/05/74      CHARACTER STRINGS IN BOX-NUM
.      E M SCHLOSSER      LEC      12/27/79      CHANGE TO BITSTRINGS IN LOOK41
```

METHOD  
-----

```
.      EXPAND BITSTRING FOR LINE OF DIGIT INTO CHARACTER STRING. USING
.      SPECIFIED FOREGROUND AND BACKGROUND CHARACTERS.
```

MACHINE-DEPENDENT CODE  
-----

```
.      WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 8-BIT
.      FIELDATA CHARACTERS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.
.      BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.
.      IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.
.      DIFFERENT COMPILERS (EO.. UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.
```

EXTERNAL REFERENCES  
-----

```
.      NONE.
```

EXCEPTIONS  
-----

```
.      1. THE FOLLOWING CONDITIONS GENERATE THE RESULT SHOWN:
```

CONDITION	VALUE OF LBOXD
IDIGIT < 0	UNDEFINED
IDIGIT > 9	UNCHANGED
LINE < 1	UNCHANGED
LINE > 12	UNCHANGED

GLOBAL DECLARATIONS  
-----



LOOK41  
002

[illegible]

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**L80X41  
003**

	OF	1.1.0.0.0.0.0.0.0.1.1.	:
		0.0.0.0.0.0.0.0.1.1.1.	:
		0.0.0.0.0.0.1.1.1.0.0.	:
	OF	0.0.0.0.0.0.0.1.1.1.0.0.	:
		0.0.0.0.0.0.0.0.0.1.1.1.	:
		1.1.0.0.0.0.0.0.0.0.1.1.	:
	OF	1.1.1.0.0.0.0.0.0.1.1.1.	:
		0.1.1.1.1.1.1.1.1.1.0.	:
		0.0.0.1.1.1.1.1.1.0.0.0.	:
FOUR	OF	0.0.0.0.0.1.1.0.0.1.1.0.	:
		0.0.0.0.0.1.1.0.0.1.1.0.	:
		0.0.0.0.1.1.0.0.0.1.1.0.	:
	OF	0.0.0.1.1.0.0.0.0.0.1.1.0.	:
		0.0.1.1.0.0.0.0.0.1.1.0.	:
		0.1.1.1.1.1.1.1.1.1.1.	:
	OF	1.1.1.1.1.1.1.1.1.1.1.1.	:
		0.0.0.0.0.0.0.0.0.1.1.0.	:
		0.0.0.0.0.0.0.0.0.1.1.0.	:
	OF	0.0.0.0.0.0.0.0.0.1.1.0.	:
		0.0.0.0.0.0.0.0.0.1.1.0.	:
		0.0.0.0.0.0.0.0.0.1.1.0.	:
FIVE	OF	1.1.1.1.1.1.1.1.1.1.1.1.	:
		1.1.1.1.1.1.1.1.1.1.1.1.	:
		1.1.0.0.0.0.0.0.0.0.0.0.	:
	OF	1.1.0.0.0.0.0.0.0.0.0.0.	:
		1.1.0.0.0.0.0.0.0.0.0.0.	:
		1.1.1.1.1.1.1.1.1.1.0.0.	:
	OF	1.1.1.1.1.1.1.1.1.1.1.0.	:
		0.0.0.0.0.0.0.0.0.0.1.1.	:
		0.0.0.0.0.0.0.0.0.0.1.1.	:
	OF	1.1.0.0.0.0.0.0.0.0.1.1.	:
		0.1.1.1.1.1.1.1.1.1.1.0.	:
		0.0.1.1.1.1.1.1.1.1.0.0.	:
SIX	OF	0.0.0.1.1.1.1.1.1.0.0.0.	:
		0.1.1.1.1.1.1.1.1.1.1.0.	:
		1.1.1.0.0.0.0.0.0.0.1.1.	:
	OF	1.1.0.0.0.0.0.0.0.0.0.0.	:
		1.1.0.0.0.0.0.0.0.0.0.0.	:
		1.1.0.1.1.1.1.1.1.1.0.0.	:
	OF	1.1.1.1.1.1.1.1.1.1.1.0.	:
		1.1.0.0.0.0.0.0.0.0.1.1.	:
		1.1.0.0.0.0.0.0.0.0.1.1.	:
	OF	1.1.0.0.0.0.0.0.0.0.1.1.	:
		0.1.1.1.1.1.1.1.1.1.1.0.	:
		0.0.0.1.1.1.1.1.1.0.0.0.	:
SEVEN	OF	1.1.1.1.1.1.1.1.1.1.1.1.	:
		1.1.1.1.1.1.1.1.1.1.1.1.	:
		0.0.0.0.0.0.0.0.0.1.1.1.	:
	OF	0.0.0.0.0.0.0.0.0.1.1.0.	:
		0.0.0.0.0.0.0.1.1.1.0.0.	:
		0.0.0.0.0.0.1.1.1.0.0.0.	:
	OF	0.0.0.0.0.1.1.1.0.0.0.	:

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

LBOX41  
884

```

      0.0.0.0.1.1.1.0.0.0.0.0.  :
      0.0.0.1.1.1.0.0.0.0.0.0.  :
      0.0.1.1.1.0.0.0.0.0.0.0.  :
      0.1.1.1.0.0.0.0.0.0.0.0.  :
      1.1.1.0.0.0.0.0.0.0.0.0.  :
      1.1.1.0.0.0.0.0.0.0.0.0.  :

EIGHT      BF      0.0.0.1.1.1.1.1.1.0.0.0.  :
      0.0.1.1.1.1.1.1.1.1.0.0.  :
      0.1.1.0.0.0.0.0.0.1.1.0.  :
      0.1.1.0.0.0.0.0.0.1.1.0.  :
      0.0.1.1.0.0.0.0.1.1.0.0.  :
      0.0.0.1.1.1.1.1.1.0.0.0.  :
      0.1.1.1.1.1.1.1.1.1.1.0.  :
      1.1.1.0.0.0.0.0.0.1.1.1.  :
      1.1.0.0.0.0.0.0.0.0.1.1.  :
      1.1.1.0.0.0.0.0.0.1.1.1.  :
      0.1.1.1.1.1.1.1.1.1.1.0.  :
      0.0.1.1.1.1.1.1.1.1.0.0.  :

NINE      BF      0.0.0.1.1.1.1.1.1.0.0.0.  :
      0.1.1.1.1.1.1.1.1.1.1.0.  :
      1.1.0.0.0.0.0.0.0.0.1.1.  :
      1.1.0.0.0.0.0.0.0.0.1.1.  :
      0.1.1.1.1.1.1.1.1.1.1.1.  :
      0.0.1.1.1.1.1.1.1.0.1.1.  :
      0.0.0.0.0.0.0.0.0.0.1.1.  :
      0.0.0.0.0.0.0.0.0.0.1.1.  :
      1.1.0.0.0.0.0.0.0.0.1.1.  :
      0.1.1.1.1.1.1.1.1.1.1.0.  :
      0.0.0.1.1.1.1.1.1.0.0.0.  :

.
.
.  PROCEDURE
.  -----
.
S(01) . 1-BANK
LBOX41*      LMA      A1.*1.X11      . A1 := /101017/
              LA       A3.*4.X11      . A3 := LINE

CHKDIOIT      TO.U      A1.10          . IF /101017/ >= 10 ...
              J         RETURN        . ... THEN RETURN

CNKLINE      ANA.U      A3.13          . A3 := (LINE-13)
              JZ        A3.RETURN     . IF LINE = 13, THEN RETURN
              JP        A3.RETURN     . IF LINE > 13, THEN RETURN
              AA.U      A3.12          . A3 := (LINE-1)
              TZ        A3           .
              JN        A3.RETURN     . IF LINE < 1, THEN RETURN

FORBAK      LA.S1      A4.*2.X11      . A4 := FOREGROUND CHARACTER
              LA.S1      A5.*3.X11      . A5 := BACKGROUND CHARACTER

FINDBS      SZ         A2             . DIVISION CONING1
              LSSL      A1.2          . A1 := /101017/ * 4 NOS OF BS PER DIOIT

```

**DAH PACKAGE APPENDIX N  
UTILITY ROUTINES**

**LBOX41  
005**

.	DI.U	A2.3	. A2 := (LINE-1)/3
.	AA	A2.A1	. A3 := MOD((LINE-1),3)
.	MS1.U	A3.12	. A2 := STARTING WORD OF BITSTRING
.			. A3 := STARTING BIT OF BITSTRING
LOADBS	LA	A2.ZERO.A2	. A2 := BITSTRING
.	LSSL	A2.0.A3	. ALIGN BITSTRING IN A2
.	LA.U	A3.11	. 12 REPETITIONS (FOR 12 CHARS)
CHR4BIT	SA	A0.TEMP	. GET WORD FROM CHRSTRING
.	JP	A2.BITOFF	. JUMP IF BIT IN BITSTRING IS OFF
BITON	SA.S1	A4.TEMP	. BIT IS ON -- INSERT FOREGROUND CHAR
.	J	ROTATE	.
BITOFF	SA.S1	A5.TEMP	. BIT IS OFF -- INSERT BACKGROUND CHAR
ROTATE	LA	A0.TEMP	. REPLACE WORD IN CHRSTRING
.	LSSC	A2.1	. SHIFT BITSTRING TO LEFT BY 1 BIT
.	LOSC	A0.6	. SHIFT CHRSTRING TO LEFT BY 1 CHAR
.	J0D	A3.CHR4BIT	. REPEAT 12 TIMES
.	OS	A0.0.X11	. LBOXD := A1&A2
RETURN	J	6.X11	.
		END	

SUBROUTINE LDREGB 8 LOAD REGISTRATION PARAMETERS FROM UNIT 8 ONLY

```

C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      12/09/73      ORIGINAL CODE
C      E H SCHLOSSER      LEC      12/28/79      SUPPORT MERGED
C
C
C METHOD
C -----
C
C      TEMPORARILY SAVE COMMON VALUES OF CRITICAL REGISTRATION PARAMETERS.
C      IF UNIT 8 EXISTS, LOAD ALL REGISTRATION PARAMETERS INTO COMMON FROM
C      UNIT 8 AND COMPARE THE SAVED VALUES OF CRITICAL REGISTRATION PARAMETERS
C      WITH THEIR NEWLY-LOADED COMMON VALUES.
C      OTHERWISE LOAD NOMINAL REGISTRATION PARAMETERS & MODIFY THEM BASED ON
C      ESTIMATED SCENE CENTER PREVIOUSLY LOADED INTO COMMON BY OPEN ROUTINES.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NAMELIST I/O
C
C
C EXTERNAL REFERENCES
C -----
C
C      ERCSF      8 SUBMIT EXEC-8 CONTROL STATEMENT
C      OCONST      8 LOAD GEOMETRIC CONSTANTS
C      MDNOTE      8 PRINT/COUNT/LOG 'NOTE' DIAGNOSTIC MESSAGE
C      MDWARN      8 PRINT/COUNT/LOG 'WARNING' DIAGNOSTIC MESSAGE
C      MDFATL      8 PRINT/COUNT/LOG 'FATAL ERROR' DIAGNOSTIC MESSAGE
C      U40         8 UTM (OR STM) COORDINATES FOR GEOGRAPHIC COORDINATES
C
C
C EXCEPTIONS
C -----
C
C      1. IF THE CALL TO THIS ROUTINE IS TO LDREGB (RATHER THAN LDREON), THEN
C      ANY MISMATCH BETWEEN THE VALUE IN COMMON AND THE VALUE FROM UNIT 8 OF
C      A CRITICAL REGISTRATION PARAMETER RESULT IN A FATAL ERROR.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST      8 COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
C      INCLUDE KOMNER.LIST      8 COMMON ERIS SCENE PARAMETERS
C      INCLUDE KOMFIT.LIST      8 COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C      INCLUDE TRFORM.LIST      8 DEFINE COORDINATE TRANSFORMATION FUNCTIONS
C      INCLUDE LSTLUB.LIST      8 NAMELIST I/O SPECIFICATIONS
C      INCLUDE FACBIT.LIST      8 MNEMONICS FOR FACILITY REQUEST STATUS BITS
C

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

LDREG0  
002

```

C
C LOCAL DECLARATIONS
C -----
C
      INTEGER N           & WHERE TO GET REGISTRATION PARAMETERS FROM
      LOGICAL ATTDEF      & TRUE IF ATTITUDE (PITCH & ROLL) DEFINED. ELSE FALSE
      INTEGER KSTATOE(2)  & SATELLITE & GEOMETRY (CHARACTER STRING)
C
C
C PROCEDURE
C -----
C
      CALL TRACE('LDREG0')
      N=0           & REGISTRATION PARAMETERS MUST BE ON UNIT 0
      GO TO 100
C
C
C
C
C
      ENTRY LDREGN  & LOAD REGISTRATION PARAMETERS FROM BEST AVAILABLE SOURCE
      -----
C
      CALL TRACE('LDREGN')
      N=0           & IF PROPER REG PARAMETERS NOT ON UNIT 0. THEN USE NOMINAL ONES
C
C
C SAVE CRITICAL REGISTRATION PARAMETERS
C
      100 NERT1=NERTS(1)
          NERT2=NERTS(2)
          NERT3=NERTS(3)
          NERL1=NERLIN
          NERSA=NERSAM
          CTRL1=CTRLIN
          CTRSA=CTRSAM
          CTRLA=CTRLAT
          CTRL0=CTRLON
          PITDE=PITDE0
          ROLDE=ROLDE0
          NER0E=NER0EO
C
C
C CHECK IF UNIT 0 IS ASSIGNED
C
      CALL ERCSF(NA0,'BASO.T 0. . ')
      IF(ALREDY(NA0).EQ.1) GO TO 200
      CALL ERCSF(NA0,'FREE 0. . ')
      IF(N.EQ.0) CALL M0FATL('NO REGISTRATION PARAMETERS FROM CONTROL')
      GO TO 500
C
C
C LOAD SCENE NUMBER FROM UNIT 0 & 6 WITH SCENE NUMBER FROM MSS
C
      200 READ(0,LSTNER.END-280)

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

LDR08  
003

```

      IF(NERT1.EQ.0) GO TO 240      8 NO MSS DATA
      IF(NERTS(2).EQ.0) NERTS(2)=NERT2      8-8-8-8 TEMP PATCH 8-8-8-8
      IF(NERTS(3).EQ.0) NERTS(3)=NERT3      8-8-8-8 TEMP PATCH 8-8-8-8
      IF((NERT1.NE.NERTS(1)).OR.
      8 (NERT2.NE.NERTS(2)).OR.
      8 (NERT3.NE.NERTS(3))) GO TO 400

C
C
C LOAD REMAINING EXACT REGISTRATION PARAMETERS FROM UNIT 8
C
      240 READ(8,LSTORB.END=280)
      READ(8,LSTSCN.END=280)
      READ(8,LSTFIT.END=300)
      GO TO 300
      280 CALL MDFATL(
      * 'BAD REGISTRATION PARAMETERS FROM CONTROL')
      GO TO 400      8 LOAD NOMINAL PARAMETERS

C
C
C VERIFY EXACT REGISTRATION PARAMETERS
C
      300 IF(NERLI.NE.NERLIN) CALL MDFATL(
      * 'LINES/SCENE DOES NOT MATCH CONTROL NETWORK')
      IF(NERSA.NE.NERSAM) CALL MDFATL(
      * 'SAMPLES/SCENE DOES NOT MATCH CONTROL NETWORK')
      IF(CTRLI.NE.CTRLIN) CALL MDFATL(
      * 'CENTER LINE DOES NOT MATCH CONTROL NETWORK')
      IF(CTRSA.NE.CTRSAM) CALL MDFATL(
      * 'CENTER SAMPLE DOES NOT MATCH CONTROL NETWORK')
      IF(ABS(CTRLAT-CTRLA).GT.0.5) CALL MDFATL(
      * 'SCENE LATITUDE DOES NOT MATCH CONTROL NETWORK')
      IF(ABS(CTRLON-CTRLO).GT.0.5) CALL MDFATL(
      * 'SCENE LONGITUDE DOES NOT MATCH CONTROL NETWORK')
      ATTOEF=((ABS(PITDE).LE.99.).AND.(ABS(ROLDE).LE.99.))
      IF((ATTOEF).AND.(ABS(PITDEG-PITDE).GT.0.20)) CALL MDFATL(
      * 'PITCH DOES NOT MATCH CONTROL NETWORK')
      IF((ATTOEF).AND.(ABS(ROLDEG-ROLDE).GT.0.20)) CALL MDFATL(
      * 'ROLL DOES NOT MATCH CONTROL NETWORK')
      IF(NERGE.NE.NERGE0) CALL MDFATL(
      * 'GEOMETRY DOES NOT MATCH CONTROL NETWORK')

C
C
C CHECK EXACT REGISTRATION PARAMETERS TO SEE IF CONTROL NETWORK ADJUSTMENT IS OK
C
      IF(NCTLPT.LT.8) CALL MDFATL('FEWER THAN 8 CONTROL POINTS')
      IF(PCYCTL.LT.10.) CALL MDFATL('INADEQUATE CONTROL COVERAGE')
      IF(RMSMET.GT.500) CALL MDFATL('RMS ADJUSTMENT ERROR TOO LARGE')
      IF(NOTOTL.EQ.0) CALL MDNOTE(
      * 'EXACT REGISTRATION BASED ON CONTROL NETWORK')
      GO TO 900

C
C
C RESTORE SCENE NUMBER
C
      400 IF(N.EQ.8) CALL MDFATL(
      * 'SCENE NUMBER DOES NOT MATCH CONTROL NETWORK')

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

LDRE00  
000

```

      NERTS(1)=NERT1
      NERTS(2)=NERT2
      NERTS(3)=NERT3
C
C
C LOAD NOMINAL REGISTRATION PARAMETERS VIA SYSIN RUNSTREAM
C
500 IF(NERSAT(1).EQ.0) CALL OCONST
      CALL MOVCS(1,KSATOE,(1),(6), NERSAT(1),(6), ' ')
      CALL PUTCHR(KSATOE,(7), ' ')
      CALL MOVCS(1,KSATOE,(8),(5), NERSAT(1),(3), ' ')
      CALL SYSADD(LOCFIL,'DAM','REG-NOM',KSATOE)
      IF(LOCFIL.GT.0) GO TO 550
      CALL MCFATL('PROGRAM ERROR -- NO NOMINAL REGISTRATION PARAMETERS')
      GO TO 900
550 READ(5,LSTSCN,END=580)
      READ(5,LSTFIT,END=580)
      CALL SYSGET(INSTAT,1,DUMMY,LENGTH)      3 GET END-OF-ADD
      IF(INSTAT.EQ.'EOA') GO TO 700
580 CALL MCFATL(
      - 'PROGRAM ERROR -- NOMINAL REGISTRATION PARAMETERS BAD')
C
C
C MODIFY NOMINAL COEFFICIENTS BASED ON ESTIMATED SCENE CENTER FROM TAPE HEADER
C
700 IF(ABS(STMCHD-CTRLON).GT.4.) STMCHD=CTRLON
      CALL U40(STMCHD,STMN, CTRLAT,CTRLON,STMCHD)
      CORLIN=CORL4A(CTRLIN,CTRSAM)
      CORSAM=CORS4A(CTRLIN,CTRSAM)
      DSTME=STME-STME4C(CORLIN,CORSAM)
      DSTMN=STMN-STMN4C(CORLIN,CORSAM)
      CORSTM(3)=CORSTM(3)+DSTMN
      CORSTM(8)=CORSTM(8)+DSTME
      CALL REVERT(CORSTM,STMCOR)
      CALL MOWARN(
      - 'NOMINAL REGISTRATION BASED ON ESTIMATED SCENE CENTER')
C
C
C RETURN
C
900 RETURN
      END

```



DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

LOCDSF  
001

INTEGER FUNCTION 8 LOCATE DISK SYMBOLIC FILE OR ELEMENT  
0 8 LOCATION WITHIN FILE: <0 NOT FOUND  
8 LOCDSF( 8 8 FILE  
1 NAMEFIL. 8 NAME OF FILE >0 ELEMENT LOCATION  
1 NAMELT. 8 NAME OF SYMBOLIC ELEMENT  
1 NAMEVER) 8 NAME OF VERSION

```

C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      01/17/79      ORIGINAL CODE
C
C
C METHOD
C -----
C
C      CHECK IF FILE EXISTS ON DISK. THEN SEARCH FILE TABLE OF CONTENTS
C      FOR ELEMENT. ELEMENT LOCATION. REGARDLESS OF COMPUTER OR OPERATING
C      SYSTEM. IS ALWAYS RETURNED IN WORDS.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      UNIVAC EXEC 8 PROGRAM FILE NAMING CONVENTIONS AND EXECUTIVE REQUESTS.
C      UNIVAC FASTRAND SECTOR SIZE = 28 WORDS.
C
C
C EXTERNAL REFERENCES
C -----
C
C      ERFACL      8 GET FACILITIES INFORMATION ON FILE
C      ERPFS      8 SEARCH EXEC-8 PROGRAM FILE TABLE OF CONTENTS
C
C
C EXCEPTIONS
C -----
C
C      1. FILE OR ELEMENT DOES NOT EXIST.
C
C      2. FILE OR ELEMENT IS NOT SYMBOLIC.
C
C      3. FILE NOT ASSIGNED TO CURRENT RUN.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE NULCST.LIST      8 DEFINE NULL CHARACTER STRING
C
C
C LOCAL DECLARATIONS
C -----
C

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

LOCDSF  
002

```

      INTEGER NAMFIL(2),NAMELT(2),NAMVER(2)      & ARGUMENTS
      INTEGER IPFPKT(12)      & PROGRAM FILE PACKET
      INCLUDE ASMDEF.LIST

C
C
C PROCEDURE
C -----
C
C
C      CALL TRACE
C
C
C INITIALIZE FILE NAME
C
      IPFPKT(1)=NAMFIL(1)
      IPFPKT(2)=NAMFIL(2)
      IF(IPFPKT(2).EQ.NULCST) IPFPKT(2)=' '
C
C
C SEE IF FILE EXISTS ON DISK
C
      LOCDSF=-3      & (NO PROGRAM FILE)
      CALL ERFACL(IPFPKT)
      IF(ASMS1(IPFPKT(7)).LT.16) GO TO 900      & NO FILE ON DISK
C
C
C CHECK IF ELEMENT & VERSION NAMES ARE NULL
C
      LOCDSF=0      & (FILE, NOT ELEMENT)
      IF((NAMELT(1).EQ.' ') .AND. (NAMVER(1).EQ.' ')) GO TO 900      & FILE
C
C
C INITIALIZE ELEMENT & VERSION NAMES
C
      IPFPKT(3)=NAMELT(1)
      IPFPKT(4)=NAMELT(2)
      IF(IPFPKT(4).EQ.NULCST) IPFPKT(4)=' '
      IPFPKT(7)=NAMVER(1)
      IPFPKT(8)=NAMVER(2)
      IF(IPFPKT(8).EQ.NULCST) IPFPKT(8)=' '
C
C
C FIND LOCATION WITHIN DISK FILE
C
      ASMS3(IPFPKT(6))=1      & SYMBOLIC ELEMENT
      IPFPKT(11)=-1      & (NO FIND)
      CALL ERPFS(IPFPKT,ISTAT)
      LOCDSF=-1ABS(ISTAT)
      IF(ISTAT.EQ.0) LOCDSF=IPFPKT(11)*28      & UNIVAC 1100: 28 WDS / DISK SECTOR
C
C
C DONE
C
      900 RETURN
      END

```

**LOG2**  
**001**

## HISTORY

## METHOD

1. MACHINE-DEPENDENT CODE

### 4. EXTERNAL REFERENCES

## EXCEPTIONS

## GLOBAL DECLARATIONS

```

      AXRS      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
S(00) . 1-BANK
L002*  LSC      A1.*0.X11      . A2 := # OF LEFT SHIFTS TIL INIPOS
      .          .          .      STARTS IN BIT 34
      .          .
      LA.U      A0.34
      ANA       A0.A2      . A0 := (#-1) OF SIGNIF BITS IN INIPOS

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

LOG2  
002

J  
END

2.X11

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

HAPH00  
001

SUBROUTINE HAPH00( I NUNIT) & WRITE MAP HEADING ON NUNIT  
& LOGICAL UNIT TO WRITE MAP HEADING ON

```

C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      07/18/73      ORIGINAL CODE
C      E M SCHLOSSER      LEC      08/29/78      IDENTIFY COUNT PER PIXEL
C      E M SCHLOSSER      LEC      01/13/79      IDENTIFY RAD/DEN/CLA RANGE
C      E M SCHLOSSER      LEMSCO    02/07/80      UPGRADE DOCUMENTATION
C      E M SCHLOSSER      LEMSCO    05/28/80      WRITE DETECTION CHANNEL NUMBER
C
C
C METHOD
C -----
C
C      PRINT INFORMATION.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      FORMAT SPECIFICATIONS ASSUME 6 CHARACTERS PER INTEGER.
C
C
C EXTERNAL REFERENCES
C -----
C
C      NONE.
C
C
C EXCEPTIONS
C -----
C
C      NONE.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMOHV.LIST      & COMMON OUTPUT WINDOW PACKETS
C      INCLUDE WINDEF.LIST      & DEFINE STRUCTURE OF WINDOW PACKETS
C      INCLUDE KOMFIT.LIST      & COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C      INCLUDE KOMKLS.LIST      & COMMON CLASSIFICATION INFO
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER ITEMP(8)          & TEMPORARY
C      REAL RLTEMP(2)           & TEMPORARY
C
C
C PROCEDURE

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

NAPNDG  
002

```

C -----
C
C      CALL TRACE
C
C
C WRITE DETECTION CHANNEL NUMBER & RADIANCE/DENSITY/CLASS RANGE
C
C      WRITE(NUNIT,115) LINCMI
115 FORMAT(1X,/
& 6X,'DETECTION CHANNEL: '.12)
      IF(KLSTYP.EQ.'RAD') WRITE(NUNIT,125) LCVLOI.LCVHI1
125 FORMAT(
& 6X,'RADIANCE RANGE: '.J3.' TO '.J3)
      IF(KLSTYP.EQ.'DEN') WRITE(NUNIT,135) LCVLOI.LCVHI1
135 FORMAT(
& 6X,'DENSITY RANGE: '.J3.' TO '.J3)
      IF(KLSTYP.EQ.'CLA') WRITE(NUNIT,145) LCVLOI.LCVHI1
145 FORMAT(
& 6X,'CLASS RANGE:  '.J3.' TO '.J3)
C
C
C WRITE COUNT PER PIXEL
C
C      175 FORMAT(
& 6X,'COUNT PER PIXEL: '.13)
C
C
C DETERMINE COORDINATE SYSTEM USED FOR ORIGIN
C
C      IF(KSYOHM(WORIO).EQ.'SCA') GO TO 300
C      IF(KSYOHM(WORIO).EQ.'DEG') GO TO 400
C      IF(KSYOHM(WORIO).EQ.'MIN') GO TO 400      & VALID FOR SUB-WINDOW 11
C      GO TO 800
C
C
C
C SCANNER ORIGIN
C
C      300 WRITE(NUNIT,315) MSAOHM(WLIN.WORIO).MSAOHM(WSAM.WORIO)
315 FORMAT('0'.5X,'ORIGIN: '.14.' SCAN LINE. '.14.' SAMPLE')
C      GO TO 800
C
C
C
C GEOGRAPHIC ORIGIN
C
C      400 CALL RL2ISX(GEDOHM(WLAT.WORIO)+.00001,ITEMP(1),3,RLTEMP(1))
C      CALL RL2ISX(GEDOHM(WLON.WORIO)+.00001,ITEMP(4),3,RLTEMP(2))
C      WRITE(NUNIT,415)
C      & (ITEMP(N),N=1,3),RLTEMP(1),
C      & (ITEMP(N),N=4,6),RLTEMP(2)
415 FORMAT('0'.5X,'ORIGIN: '.
& 14.' '.J2.' '.J2.F2.1.' LAT. ',
& 14.' '.J2.' '.J2.F2.1.' LON')
C      GO TO 800
C
C
C
C UTM ORIGIN

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

HAFH00  
003

```

C
000 WRITE(NUNIT,015) UTHOMM(MEA,WOR10),UTHOMM(WNO,WOR10)
015 FORMAT('C',5X,'ORIGIN: ',3P,F0.2,' KM E. ',F0.2,' KM N')
    GO TO 000

C
C
C WRITE SCALE
C
000      IF(IRFD.LT.100000) WRITE(NUNIT,015) IRFD
015      FORMAT(1X,/
    3 5X,'MAP SCALE: ',1/'1'/.15)
      IF(IRFD.GE.100000) WRITE(NUNIT,025) IRFD
025      FORMAT(1X,/
    3 5X,'MAP SCALE: ',1/'1'/.15)

C
C
C WRITE REGISTRATION INFORMATION
C
      WRITE(NUNIT,035) STNCHD
035      FORMAT(1X,/
    4 5X,'PROJECTION: TRANSVERSE MERCATOR'/
    5 5X,'SPHEROID: CLARKE 1866'/
    6 5X,'CENTRAL MERIDIAN: ',F0.4,' DEGREES')
      WRITE(NUNIT,045) NCTLPT,PCTCTL,RMSMET
045      FORMAT(1X,/
    7 5X,'CONTROL: ',13,' POINTS COVERING ',F4.
    7      ' PERCENT OF ERTS SCENE'/
    8 5X,'ROOT MEAN SQUARE ERROR: ',F5,' METERS'///)
      RETURN
      END

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**MATPRT  
001**

```

SUBROUTINE MATPRT( I LOGICAL UNIT NUMBER
I REAL SINGLE PRECISION MATRIX
I NAME OF ROWS (8-CHARACTER STRING)
I NRUSE,NCUSE. I NUMBER OF ROWS & COLUMNS USED
I NRDIM,NCDIM) I NUMBER OF ROWS & COLUMNS DIMENSIONED
-----
C
C
C HISTORY
C -----
C
C      M L BROWN          LEC      01/09/78      ORIGINAL CODE
C
C METHOD
C -----
C
C      PRINT WITH F10.5 FORMAT, ONE ROW PER LINE.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      NONE.
C
C EXCEPTIONS
C -----
C
C      1. ANY ROW WITH MORE THAN 6 COLUMNS IS PRINTED ON MORE THAN 1 LINE.
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C LOCAL DECLARATIONS
C -----
C
C      REAL RMAT(NRDIM,NCDIM) I ARGUMENT
C      INTEGER IROW          I ROW NUMBER TO PRINT
C      INTEGER NR            I CURRENT ROW NUMBER
C      INTEGER NC            I CURRENT COLUMN NUMBER
C
C
C PROCEDURE

```



GAM PACKAGE APPENDIX M  
UTILITY ROUTINES

NATPRT  
002

```
C -----  
C  
C  
C IF ONLY 1 ROW IS TO BE PRINTED & WITHOUT NAME. SUPRESS ROW NUMBER  
C  
    IROW=0  
    IF((NRUSE.EQ.1).AND.(NARROW.EQ.' ')) IROW=9999  
C  
C  
C PRINT MATRIX  
C  
    DO 150 NR=1,NRUSE  
        IROW=IROW+1  
        WRITE(LUNIT,125) NARROW,IROW,(ALMAT(NR,NC),NC=1,NCUSE)  
125    FORMAT(1X,A6,12,(110,SF10.5))  
150 CONTINUE  
C  
    WRITE(LUNIT,165)  
165 FORMAT(1X)  
C  
    RETURN  
END
```

MDL 00  
001

1 MSG01.MSG02.MSG03.MSG04.MSG05.MSG06.V0107.V0108) 0 0 TO 6 MESSAGE(S)

E M SCHLOSSER	LEC	03/28/73	MULTIPLE ENTRY POINT CODE IN MOLOG
E M SCHLOSSER	LEC	01/03/78	REV & SEPARATE ROUTINES
E M SCHLOSSER	LEC	09/22/79	COMBINE. ADD QUEUED & MULTIPLE MSGS
E M SCHLOSSER	LEC	12/05/79	DON'T QUEUE IF TRACE IS ON

THIS MULTIPLE ENTRY POINT TRANSFORM PROCESSES ALL DIAGNOSTICS FOR THE DAM PACKAGE.  
DETERMINE HOW MANY OF OPTIONAL MESSAGES ARE PRESENT. PUT PREFIX & MESSAGE(S) INTO LOG BUFFER. IF BUFFER IS NOT EMPTY, THEN SAVE IT IN QUEUE FILE IF MSG1 STARTS WITH '('. ELSE PRINT IT IMMEDIATELY.  
AS APPROPRIATE TO EACH ENTRY POINT, WRITE BUFFER INTO PROPER SECTOR OF LOG FILE. ADJUST LOG FILE SECTOR POINTERS, AND UPDATE DIAGNOSTIC COUNTERS.

ENTRY	PRT/QUE MSG(\$)	LOG MSG(\$)	CHANGE NDWARN	CHANGE NOFATL	CHANGE NOCLRW	CHANGE NOCLRF
-----	-----	-----	-----	-----	-----	-----
MDLOG	NO	YES	NO	NO	NO	NO
MONOTE	YES	YES	NO	NO	NO	NO
MDWARN	YES	YES	+1	NO	NO	NO
MOFATL	YES	YES	NO	+1	NO	NO
MDCLRW	YES	NO	0	NO	+NDWARN	NO
MDCLRF	YES	YES	NO	0	NO	+NOFATL

ANY QUEUED DIAGNOSTICS WILL BE AUTOMATICALLY DE-QUEUED AND PRINTED ON THE NEXT CALL TO READ5.

UNIVAC FORTRAN V RETURN K  
UNIVAC EXEC-B I/O PACKETS

```

C
C EXTERNAL REFERENCES
C -----
C
C      ARGRET      & DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR
C      CBINIT      & INITIALIZE CHARACTER BUFFER
C      CCHYST      & CHARACTER BUFFER FOR CHARACTER STRING
C      ERION       & INITIATE I/O AND WAIT FOR COMPLETION
C      INTEGER LENCST & LENGTH OF CHARACTER STRING
C      INTEGER ICE   & INTEGER-CHARACTER-EQUIVALENT
C
C
C
C EXCEPTIONS
C -----
C
C      1. PSTART MUST BE CALLED BEFORE THE FIRST CALL TO THIS ROUTINE.
C
C      2. MSG1. . . MSG8. IF PRESENT, MUST BE VARIABLE-LENGTH CHARACTER STRINGS
C         (HOLLERITH LITERALS, OR ARRAYS OF PACKED CHARACTERS TERMINATED WITH
C         NULCHR OR NULCST).
C
C      3. IF MORE THAN 8 ACTUAL MESSAGE ARGUMENTS ARE PROVIDED, THE RESULTS ARE
C         UNDEFINED, AND PROGRAM ABORT TERMINATION MAY OCCUR.
C
C      4. IF THE COMBINED LENGTH OF ALL MSG(S) PRESENT EXCEEDS 120 CHARACTERS,
C         TRUNCATION WILL OCCUR.
C
C      5. IF MSG1 IS ABSENT OR NULL, NO MESSAGE IS QUEUED, PRINTED, OR LOGGED.
C
C      6. THE DIAGNOSTIC COUNTERS NDHARN, NDFATL AND NOTOTL IN LABELLED COMMON
C         (PROC KOMXQT) MAY BE CHECKED BY OTHER ROUTINES BUT MUST NOT BE CHANGED
C         BY THEM.
C
C      7. KOMXQT AND KOMLOO MUST BE INCLUDED IN THE MAIN PROGRAM (ROOT OVERLAY
C         SEGMENT) OR ELSE THIS ROUTINE WILL NOT WORK PROPERLY.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
C      INCLUDE KOMLOO.LIST     & COMMON LOG FILE BUFFER, I/O PKT. POINTERS
C      INCLUDE KOMIO.LIST      & COMMON I/O FUNCTIONS
C      INCLUDE NULCST.LIST     & DEFINE NULL CHARACTER STRING
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER NMSG          & NUMBER OF ACTUAL MESSAGES
C      INTEGER KRETN         & RETURN K VECTOR
C
C
C PROCEDURE
C -----

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

MDLOG  
003

```

C
C
C PUT NUL PREFIX & MESSAGE(S) INTO BUFFER & DON'T PRINT OR QUEUE IT
C
    CALL TRACE('MDLOG')
    CALL ARQRET(NMSOS,KRETN)
    CALL M$UFPO(NULCST,NMSOS)
    IF(LENCST(L$UFR,12).LT.2) GO TO 900
C
C
C WRITE BUFFER TO TERMINATION SECTOR OF LOG FILE
    IOSECT(LOOPKT)=L$TERM
    IOFUNC(LOOPKT)='3C'      & WRITE
    CALL ERIOH(LOOPKT)
    GO TO 900
C
C
C
C
C ENTRY MDNOTE( & QUEUE/PRINT & LOG 'NOTE' DIAGNOSTIC MESSAGE(S)
C
C I MS01,MS02,MS03,MS04,MS05,MS06,VOID7,VOID8)      & 0 TO 6 MESSAGE(S)
C (QUEUE IF MS01 STARTS WITH '(' ELSE PRINT)
C -----
C
C PUT PREFIX & MESSAGE(S) INTO BUFFER & PRINT OR QUEUE IT
C
    CALL TRACE('MDNOTE')
    CALL ARQRET(NMSOS,KRETN)
    CALL M$UFPO(' ***NOTE:',NMSOS)
    IF(LENCST(L$UFR,12).LT.2) GO TO 900
C
C
C WRITE BUFFER TO NOTE SECTOR OF LOG FILE
C
    IF(L$NOTE.LT.4) GO TO 900
    IOSECT(LOOPKT)=L$NOTE
    IOFUNC(LOOPKT)='3C'      & WRITE
    CALL ERIOH(LOOPKT)
    GO TO 900
C
C
C
C
C ENTRY MDWARN( & QUEUE/PRINT & LOG & TALLY 'WARNING' DIAGNOSTIC MESSAGE(S)
C
C I MS01,MS02,MS03,MS04,MS05,MS06,VOID7,VOID8)      & 0 TO 6 MESSAGE(S)
C (QUEUE IF MS01 STARTS WITH '(' ELSE PRINT)
C -----
C
C INCREMENT WARNING COUNTER
C
    CALL TRACE('MDWARN')

```

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

MDL00  
004

```

      NDHARN=NDHARN+1
C
C
C OFFER HELP TO DEMAND USER IN TROUBLE
C
      IF((NDHARN.EQ.5).AND.      5 5TH WARNING
      & (NBATCH.EQ.0))      5 DEMAND TERMINAL RUN
      & CALL ERPRNT(1.4,' FOR HELP ENTER: EXPLAIN')
C
C
C PUT PREFIX & MESSAGE(S) INTO BUFFER & PRINT OR QUEUE IT WITH BELLS
C
      CALL ARQRET(NMSOS,KRETN)
      CALL MBUFPG('***WARNING:'.NMSOS)
      IF(LENCST(LBUFR,12).LT.2) GO TO 900
C
C
C WRITE BUFFER TO WARNING SECTOR OF LOG FILE
C
      IF(LOHARN.LT.4) GO TO 900
      IOSECT(LOOPKT)=LOHARN
      IOFUNC(LOOPKT)='8C'      5 WRITE
      CALL ERIOH(LOOPKT)
C
C
C SUPPRESS LOGGING OF NOTES
C
      LONOTE=0
      GO TO 900
C
C
C
C
      ENTRY MDFATL(      5 QUEUE/PRINT & LOG & TALLY 'FATAL ERROR' MESSAGE(S)
      1 MS01,MS02,MS03,MS04,MS05,MS06,VOID07,VOID08)      5 0 TO 6 MESSAGE(S)
      (QUEUE IF MS01 STARTS WITH '(' ELSE PRINT)
      -----
C
C
C INCREMENT FATAL ERROR COUNTER
C
      CALL TRACE('MDFATL')
      MDFATL=MDFATL+1
C
C
C PUT PREFIX & MESSAGE(S) INTO BUFFER & PRINT OR QUEUE IT WITH BELLS
C
      CALL ARQRET(NMSOS,KRETN)
      CALL MBUFPG('*****FATAL ERROR:'.NMSOS)
      IF(LENCST(LBUFR,12).LT.2) GO TO 900
C
C
C WRITE BUFFER TO FATAL-ERROR SECTOR OF LOG FILE
C
      IF(LOFATL.LT.4) GO TO 900

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

MDL00  
005

```

        IOSECT(LOOPKT)=LOFATL
        IOFUNC(LOOPKT)='8C'      & WRITE
        CALL ERION(LOOPKT)

C
C
C PUT FIRST FATAL ERROR MESSAGE(S) ON TAIL SHEET
C
      500 IF(NDFATL.NE.1) GO TO 600
          LBUF(1)='8LOG '
          LBUF(22)=' '
          CALL ERCSF(NDTOTL,LBUF)

C
C
C SUPPRESS LOGGING OF NOTES & WARNINGS
C
      600 LONOTE=0
          LOWARN=0
          LOFATL=LOTERM      & LOG SUBSEQUENT ERRORS IN TERMINATION SECTOR
          GO TO 900

C
C
C
C
      ENTRY MDCLRW( & QUEUE/PRINT MESSAGE(S) & CLEAR WARNING(S)
      I MS01,MS02,MS03,MS04,MS05,MS06,VOID7,VOID8)      & 0 TO 6 MESSAGE(S)
          (QUEUE IF MS01 STARTS WITH '(' ELSE PRINT)
      -----
C
C
C UPDATE WARNING COUNTERS
C
      CALL TRACE('MDCLRW')
      MDCLRW=MDCLRW+NOWARN
      NOWARN=0

C
C
C PUT PREFIX & MESSAGE(S) INTO BUFFER & PRINT OR QUEUE IT
C
      CALL ARQRET(NMS05,KRETN)
      CALL MBUFQ(' ***WARNINGS CLEARED:'.NMS05)
      GO TO 900

C
C
C
C
      ENTRY MDCLRF( & QUEUE/PRINT MESSAGE(S) & CLEAR FATAL ERROR(S)
      I MS01,MS02,MS03,MS04,MS05,MS06,VOID7,VOID8)      & 0 TO 6 MESSAGE(S)
          (QUEUE IF MS01 STARTS WITH '(' ELSE PRINT)
      -----
C
C
C UPDATE FATAL ERROR COUNTERS
C
      CALL TRACE('MDCLRF')

```

DAN PACKAGE APPENDIX H  
UTILITY ROUTINES

MDL06  
886

```

      NDCLRF=NDCLRF+NDFATL
      NDFATL=0
C
C
C PUT PREFIX & MESSAGE(S) INTO BUFFER & PRINT OR QUEUE IT
C
      CALL ARQRET(NMSOS,KRETN)
      CALL MBUFFQ(' ***FATAL ERRORS CLEARED:'.NMSOS)
      IF(LENCST(LBUFR,12).LT.2) GO TO 900
C
C
C WRITE BUFFER TO TERMINATION SECTOR OF LOG FILE
C
      IF(LOTERM.LT.4) GO TO 900
      IOSECT(LOOPKT)=LOTERM
      IOFUNC(LOOPKT)='2C'      & WRITE
      CALL ERLOW(LOOPKT)
C
C
C LOG NEXT ERROR IN DIAGNOSTIC SECTOR
C
      LOFATL=LOPCT+2
C
C
C UPDATE TOTAL DIAO COUNTER & RETURN TO NEXT INSTRUCTION OF CALLING ROUTINE
C
      900 NDTOTL=NDWARN+NDFATL+MCHECK      & PREVENTS PROCESSING WINDOW IF MCHECK NOT 0
      RETURN KRETN
C
C
C
C
C
C
C      INTERNAL
      SUBROUTINE MBUFFQ(      & JOIN PREFIX & MESSAGE(S) IN LBUFR. PRINT OR QUEUE IT
      I MDPREF.      & DIAGNOSTIC PREFIX:
      IF NULL. DON'T PRINT OR QUEUE BUFFER
      IF 1ST CHAR IS ' '. RING BELLS!!!
      I NMSOS)      & NUMBER OF MESSAGES PASSED TO MD---- ENTRY
C
C
C      INTEGER LEN1      & LENGTH OF MSG1
C
C
C ARE THERE ANY NON-NULL MESSAGES?
C
      CALL CBINIT(LBUFR)
      LEN1=0
      IF(NMSOS.GT.0) LEN1=LENCST(MSG1,1)
      IF(LEN1.EQ.0) GO TO 900      & NOTHING!!
C
C
C JOIN DIAGNOSTIC PREFIX AND MESSAGE(S) INTO SINGLE DIAGNOSTIC IN LBUFR
C
      CALL CB4CST(LBUFR. MDPREF)

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

MDL00  
007

```

                                CALL CB4CST(LBUFR, ' ')
                                CALL CB4CST(LBUFR, MS01)
IF(NMS05.07.1) CALL CB4CST(LBUFR, MS02)
IF(NMS05.07.2) CALL CB4CST(LBUFR, MS03)
IF(NMS05.07.3) CALL CB4CST(LBUFR, MS04)
IF(NMS05.07.4) CALL CB4CST(LBUFR, MS05)
IF(NMS05.07.5) CALL CB4CST(LBUFR, MS06)
IF(NMS05.07.6) CALL CB4CST(LBUFR, '**NMS05>6** IABORT!')

C
C
C QUEUE ASYNCHRONOUS DIAGNOSTIC OF 6 OR FEWER MESSAGES ON UNIT 0
C
IF(LENCST(MDPREF).EQ.0) GO TO 900      & NO PREFIX -- DON'T PRINT OR QUEUE
IF(ICE(MS01).NE.ICE(' ')) GO TO 300    & NOT ASYNCHRONOUS -- DON'T QUEUE
IF(TRACE.NE.0) GO TO 300              & TRACE IS ON -- DON'T QUEUE
IF(NMS05.07.6) GO TO 300              & WE'RE GOING TO ABORT -- DON'T QUEUE!!!
    IOSIZE(LUOPKT)=22
    IOSECT(LUOPKT)=IOSECT(LUOPKT)+1
    IOFUNC(LUOPKT)='&C'              & WRITE
    CALL ER10H(LUOPKT)
    GO TO 900

C
C
C PRINT SYNCHRONOUS DIAGNOSTIC
C
300 IF(ICE(MDPREF).EQ.ICE(' ')) CALL EAPRNT(
    & 0.2,'&&B&&B&&B&&B')      & ASCII: NUL BEL NUL BEL NUL BEL NUL BEL
    CALL ERPRNT(1.22,LBUFR)

C
C
C DONE
C
900 RETURN
END

```



MSKPIX  
001

1 NXND1) 8 PXDEF BUFFER TO BE MASKED

## C METHOD

```
C GET INTERCEPT PAIR(S) FOR WINDOW EXTERIOR AND ASSIGN NODATA THRESHOLD
C TO ALL PIXELS BETWEEN THEM.
```

### C MACHINE-DEPENDENT CODE

C UTILIZES UNIVAC FORTRAN V FUNCTION 'LOC'.

### C EXTERNAL REFERENCES

```

C      MCFATL      3 PRINT/COUNT/LOG 'FATAL ERROR' DIAGNOSTIC MESSAGE
C      WINEXT      3 COMPUTE INTERCEPT PAIR(S) FOR WINDOW EXTERIOR
C      PUTICE      3 PUT INTEGER-CHARACTER-EQUIVALENT INTO CHARACTER STRING
C      PUTBYT      3 PUT NON-NEGATIVE INTEGER INTO BYTE OF BYTE STRING
C      DOUBLE PRECISION C854CS      3 VARIABLE LENGTH CST FOR FIXED LENGTH CST

```

### C EXCEPTIONS

CONDITION	DIAGNOSTIC	ACTION
1. DIFFERENT ACTUAL ARGUMENTS USED FOR NXWDO AND NXWDI	FATAL	RETURN
2. INTERCEPT TABLE OVERFLOW	FATAL	RETURN
3. BIN TYPE IS 'NUL'		RETURN
4. BIN TYPE <> 'NUL' OR 'BYT' OR 'CHR'	FATAL	RETURN

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

MSKPIX  
002

C GLOBAL DECLARATIONS

C -----

C

INCLUDE KONXQT.LIST

% COMMON PROGRAM EXECUTION SWITCHES, COUNTERS

INCLUDE PXBDEF.LIST

% POINTERS DEFINING BUFFER STRUCTURE

C

C

C LOCAL DECLARATIONS

C -----

C

INTEGER NXWDI(1),NXWDO(1)

% ARGUMENTS

INTEGER INCEPT(16)

% INTERCEPT TABLE

INTEGER NCEP

% INTERCEPT # OF 2ND INTERCEPT IN CURRENT PAIR

INTEGER NCEPHI

% # OF INTERCEPTS (ALWAYS EVEN -- 2 \* # PAIRS)

INTEGER NODATA

% NO DATA THRESHOLD

INTEGER NBINLO

% LOW BIN NUMBER IN BUFFER TO MASK

INTEGER NBINHI

% HIGH BIN NUMBER IN BUFFER TO MASK

INTEGER NBINSO

% BIN NUMBER CORRESPONDING TO SAMPLE 0

C

C

C PROCEDURE

C -----

C

C

C CHECK THAT SAME ACTUAL ARG WAS USED FOR NXWDO & NXWDI

C

IF(LOC(NXWDO).NE.LOC(NXWDI)) CALL HDFATL(

= 'NXWDO & NXWDI NOT SAME IN MSKPIX')

C

C

C GET/CHECK INTERCEPT PAIRS

C

CALL MINEXT(INCEPT,NCEPHI,

= NXWDI(PXLINO),NXWDI(PXLSAM),NXWDI(PXHSAH))

IF(NCEPHI.GT.16) CALL HDFATL(

= 'INTERCEPT TABLE OVERFLOW IN MSKPIX')

C

C

C CHECK IF MASKING IS TO BE DONE

C

IF(NCEPHI.EQ.0) GO TO 900

IF(NDFATL.NE.0) GO TO 900

C

C

C INITIALIZE BIN OFFSET & NODATA THRESHOLD

C

NBINSO=NXWDI(PXLBIN)-NXWDI(PXLSAM)

NODATA=NXWDI(PXNODA)

C

C

C IF BIN TYPE IS 'NUL' DON'T MASK

C

IF(NXWDI(PXBINT).EQ.'NUL') GO TO 900

C

C

C ELSE IF BIN TYPE IS 'BYT' MASK BYTE-SIZED PIXELS

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

MSKPIX  
003

```

C
  IF(NXWD1(PXBINT).NE.'BYT') GO TO 300
  DO 200 NCEP=2,NCEPHI.2
    NBINLO=INCEP(NCEP-1)+NBINS0
    NBINH1=INCEP(NCEP)+NBINS0
    DO 100 NOBIN=NBINLO,NBINH1
      CALL PUTBYT(NXW00(PXBINS).(NOBIN). NODATA)
  100 CONTINUE
  200 CONTINUE
  GO TO 900

C
C
C ELSE IF BIN TYPE IS 'CHR' MASK CHARACTER-SIZED PIXELS
C
  300 IF(NXWD1(PXBINT).NE.'CHR') GO TO 600
  DO 500 NCEP=2,NCEPHI.2
    NBINLO=INCEP(NCEP-1)+NBINS0
    NBINH1=INCEP(NCEP)+NBINS0
    DO 400 NOBIN=NBINLO,NBINH1
      CALL PUTICE(NXW0C(PXBINS).(NOBIN). NODATA)
  400 CONTINUE
  500 CONTINUE
  GO TO 900

C
C
C ELSE COMPLAIN ABOUT BIN TYPE!!
C
  600 CALL MCFATL( CBS4CS(NXWD1(PXBINT).1.4).
    & ' BIN TYPE NOT SUPPORTED IN MSKPIX')

C
C RETURN TO CALLING ROUTINE
C
  900 RETURN
  END

```

. SUBROUTINE MYCONT( & MOVE CONTENTS BETWEEN SPECIFIED LOCATIONS  
. & LOCFM, & 'FROM' LOCATION  
. & LOCTO) & 'TO' LOCATION  
. -----

. HISTORY  
. -----

. E H SCHLOSSER LZC 03/17/75 DESIGN/CODE/TEST

. METHOD  
. -----

. THIS ASSEMBLER SUBROUTINE ALLOWS THE FORTRAN PROGRAMMER TO REFERENCE  
. A MACHINE WORD BY ITS LOCATION, RATHER THAN BY ITS NAME. THE FOLLOWING  
. ILLUSTRATES ITS USE IN CIRCUMVENTING THE FORTRAN RULE THAT ARRAY  
. SUBSCRIPTS MUST ALWAYS BE POSITIVE:

. DIMENSION ARRAY(1)  
. RLTEMP=ARRAY(0) & NOT VALID ... INSTEAD USE ...  
. CALL MYCONT(LOC(ARRAY)-1,LOC(RLTEMP))

. THIS SUBROUTINE SHOULD BE USED ONLY WHEN ABSOLUTELY NECESSARY, AND  
. THEN WITH EXTREME CAUTION!

. MACHINE-DEPENDENT CODE  
. -----

. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS.  
. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT COMPILERS  
. (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

. EXTERNAL REFERENCES  
. -----

. NONE

. EXCEPTIONS  
. -----

C 1. USE AT YOUR OWN RISK!

. GLOBAL DECLARATIONS  
. -----

. AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS  
. -----

**NYCONT**  
**002**

• **PROCEDURE**  
• -----

LA  
LA  
LA  
SA  
J  
END

```

. BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
A1.*0.X11      . 'FROM' LOCATION
A2.*1.X11      . 'TO' LOCATION
A0.0.A1        . GET 'FROM' CONTENTS
A0.0.A2        . STORE IN 'TO' LOCATION
3.X11          . RETURN

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**MXHLT/MATHPACK  
001**

```

SUBROUTINE MXHLT( 8 MATRIX MULTIPLY: (CHAT) = (BMAT) (AMAT)
I AMAT.      8 MULTIPLICAND MATRIX
I BMAT.      8 MULTIPLIER MATRIX
O CHAT.      8 PRODUCT MATRIX
I MRUAC.     8 NUMBER OF ROWS USED IN AMAT & CHAT
I MCRUAC.    8 NUMBER OF COLUMNS USED IN AMAT, ROWS USED IN BMAT
I NCUBC.     8 NUMBER OF COLUMNS USED IN BMAT & CHAT
I MRDAC.     8 NUMBER OF ROWS DIMENSIONED IN AMAT & CHAT
I MRDB.      8 NUMBER OF ROWS DIMENSIONED IN BMAT
-----
C
C
C
.....
..... UNIVAC 1100 MATHPACK ROUTINE .....
..... SOURCE CODE IS NOT AVAILABLE .....
.....

```

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

NEOPIC  
001

```

SUBROUTINE NEOPIC( 8 CONVERT DIGITAL PICTURE FROM POSITIVE TO NEGATIVE
U IPIX.          8 1 PIXEL/WORD:
C                M1 = MEAN VALUE (SIGNED)
C                M2 = COUNT OF ELEMENTS DEFINING MEAN (UNSIGNED)
C 1 NCOL.        8 NUMBER OF COLUMNS IN DIGITAL PICTURE
C 1 NROW)        8 NUMBER OF ROWS IN DIGITAL PICTURE
C                !!!LAST 2 ROWS ARE SCRATCH SPACE FOR ROUTINE!!!
C -----
C (E M SCHLOSSER)
C
C
C DIMENSION IPIX(NCOL,NROW)
C INCLUDE ASMDEF.LIST
C DEFINE KPIX(1,J)=ASHMR(IPIX(1,J)) 8 COUNT OF ELEMENTS DEFINING MEAN
C DEFINE MPXPUT(1,J)=ASHM1(IPIX(1,J)) 8 ASSIGN MEAN VALUE (SIGN EXTENDED)
C DEFINE MPXGET(1,J)=IPIX(1,J)/2**10 8 RETRIEVE MEAN VALUE (SIGN EXTENDED)
C CALL TRACE
C
C
C C INITIALIZE
C
C MINOLD=2**10-1 8 MAXIMUM UNSIGNED HALF-WORD INTEGER
C MINNEW=2**10-1
C NRMAL=NROW-2 8 LAST 2 ROWS ARE SCRATCH SPACE -- NOT PICTURE!!!
C
C
C C REFLECT MEANS
C
C DO 250 NC=1,NCOL
C DO 200 NR=1,NRMAL
C MINOLD=MIND(MINOLD,MPXGET(NC,NR))
C MINNEW=MIND(MINNEW,-MPXGET(NC,NR))
C MPXPUT(NC,NR)=MPXGET(NC,NR)
C 200 CONTINUE
C 250 CONTINUE
C
C
C C COMPUTE REQUIRED BIAS
C
C HNBAS=MINOLD-MINNEW
C
C
C C APPLY BIAS (TO PRESERVE ORIGINAL RANGE)
C
C DO 450 NC=1,NCOL
C DO 400 NR=1,NRMAL
C 400 MPXPUT(NC,NR)=MPXGET(NC,NR)+HNBAS
C 450 CONTINUE
C
C
C RETURN
C END

```

ORIGINAL PAGE IS  
OF POOR QUALITY

NTABS/DAM 8 I/O UNIT NUMBER TABLE  
-----

HISTORY  
-----

E M SCHLOSSER	LEC	11/05/73	ORIGINAL CODE
E M SCHLOSSER	LEC	05/16/74	RECALL FILE
E M SCHLOSSER	LEC	12/16/78	MACRO COMMAND EDIT FILES
E M SCHLOSSER	LEC	02/27/79	*DAMPLY-9
E M SCHLOSSER	LEC	08/07/79	DIAGNOSTIC QUEUE FILE

METHOD  
-----

THIS MODULE, WHEN ASSEMBLED, GENERATES A UNIT NUMBER TABLE UTILIZED BY THE UNIVAC FORTRAN V I/O ROUTINES. THE UNIT NUMBER ASSIGNMENTS GIVEN HERE APPLY TO ALL FILES IN THE DAM PACKAGE, REGARDLESS OF WHETHER FORTRAN V I/O IS PERFORMED ON THEM OR NOT. THIS MODULE MUST BE IN-ED IN THE MAP SYMBOLIC ELEMENT USED TO COLLECT EACH DAM PACKAGE PROGRAM; THIS WILL CAUSE IT TO BE USED INSTEAD OF THE INSTALLATION DEFAULT NTABS.

MACHINE-DEPENDENT CODE  
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT COMPILERS (EG., UNIVAC ASCII FORTRAN) AND DIFFERENT MACHINES.

EXTERNAL REFERENCES  
-----

NONE.

EXCEPTIONS  
-----

NONE.

GLOBAL DECLARATIONS  
-----

NONE.

C-3



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

NTABS/DAN  
002

```

.      0  DIAGNOSTIC QUEUE FILE
.      1  LOG FILE
.      2  TAPE OUTPUT
.      3  TAPE INPUT
.      4  TEMPORARY RECALL FILE FOR CARD-READER/TERMINAL INPUT
.      5  CARD-READER/TERMINAL INPUT
.      6  LINE-PRINTER/TERMINAL OUTPUT
.      8  TEMPORARY FASTRAND FILE FOR REGISTRATION PARAMETERS FROM CONTROL
.      9  ALTERNATE PRINT FILE *DAMPLT-9
.     10-19 ALTERNATE PRINT FILES *DAMPRT-0 THRU *DAMPRT-9
.      20  TEMPORARY FASTRAND SCRATCH FILE
.     21-24 DETECTION FILES *DANDET-1 THRU *DANDET-4
.     25-29 RESERVED FOR FUTURE DETECTION FILES
.      30  REREAD
.      31  CARD PUNCH
.     40-49 TEMPORARY MACRO COMMAND EDIT FILES

```

```

NSTAB 50.1.1.1.1.1.1
      6      : PRINT UNIT
      31     : CARD PUNCH UNIT
      5      : CARD READER UNIT
      30     : REREAD UNIT
9.10.11.12.13.14.15.16.17.18.19 . ALTERNATE PRINT UNITS
END .

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

NULSUB  
001

```
.      SUBROUTINE NULSUB  & DO ABSOLUTELY NOTHING!!  
.      -----  
.      (E H SCHLOSSER)  
.      S(01)  
NULSUB*      AXRS  
              J  
              END      I.X11      . RETURN
```

ORIGINAL PAGE IS  
OF POOR QUALITY

**UNCLASSIFIED**  
**001**

## HISTORY

## METHOD

## MACHINE-DEPENDENT CODE

## EXTERNAL REFERENCES

## EXCEPTIONS

## GLOBAL DECLARATIONS

## LOCAL DECLARATIONS

**N-174**

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**NVIATO  
002**

<b>WBPKT</b>	♦	<b>'NVIATO'</b>	. SUBROUTINE NAME
	♦	<b>0</b>	. SAVE LOCATION FOR RETURN ADDRESS
<b>ERRVIA</b>		<b>'NVIATO - BAD NANVIA</b>	
<b>ERRTO</b>		<b>'NVIATO - BAD NAMTO'</b>	

**. PROCEDURE**  
-----

**S(01) . I-BANK**

<b>NVIATO*</b>	<b>LA.U</b>	<b>A0.*0.X11</b>	. 1ST ARG = NANVIA
	<b>LA.U</b>	<b>A1.*1.X11</b>	. 2ND ARG = NAMTO
	<b>LA.S1</b>	<b>A2.0.A0</b>	. FIRST INSTRUCTION OF NANVIA
	<b>LA.S1</b>	<b>A3.0.A1</b>	. FIRST INSTRUCTION OF NAMTO
	<b>SA.H2</b>	<b>A0.VIA</b>	. 'VIA' ADDRESS
	<b>SA.H2</b>	<b>A1.TO</b>	. 'TO' ADDRESS
	<b>JZ</b>	<b>A2.BADVIA</b>	. ILLEGAL OP CODE (00)
	<b>JZ</b>	<b>A3.BADTO</b>	. ILLEGAL OP CODE (00)
	<b>ANA.U</b>	<b>A2.077</b>	
	<b>ANA.U</b>	<b>A3.077</b>	
	<b>JZ</b>	<b>A2.BADVIA</b>	. ILLEGAL OP CODE (OCTAL 77)
	<b>JZ</b>	<b>A3.BADTO</b>	. ILLEGAL OP CODE (OCTAL 77)
<b>RETURN</b>	<b>J</b>	<b>3.X11</b>	. NORMAL RETURN
<b>BADVIA</b>	<b>PSRINT</b>	<b>(PF 1.4.ERRVIA)</b>	
	<b>J</b>	<b>WALKBACK</b>	
<b>BADTO</b>	<b>PSRINT</b>	<b>(PF 1.3.ERRTO)</b>	
<b>WALKBACK</b>	<b>LR</b>	<b>R3.WBPKT</b>	. NAME OF THIS SUBROUTINE
	<b>SLJ</b>	<b>NERRS</b>	. INITIATE FORTRAN V WALKBACK
	♦	<b>2</b>	. NUMBER OF ARGUMENTS

**ENTRY VIATO 3 CALL VIA SUBROUTINE NANVIA TO SUBROUTINE NAMTO**  
-----

**. NOTE: NANVIA & NAMTO MUST BE SPECIFIED BY A CALL TO NVIATO, BEFORE  
CALLING VIATO. THIS CALL TO NVIATO MAY BE RECURSIVE (FROM A  
SUBROUTINE CURRENTLY CALLED BY VIATO).**

<b>VIATO*</b>	<b>SX.H2</b>	<b>X11.WBPKT+1</b>	. SAVE RETURN ADDRESS
<b>VIA</b>	<b>LNJ</b>	<b>X11.0</b>	. CALL NANVIA
<b>TO</b>	♦	<b>0</b>	. ONE ARGUMENT: ADDRESS OF NAMTO
	♦	<b>(WBPKT)</b>	. WALKBACK WORD
	<b>LX.H2</b>	<b>X11.WBPKT+1</b>	. RESTORE RETURN ADDRESS
	<b>J</b>	<b>1.X11</b>	. RETURN

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

NVIATO  
003

. THE FOLLOWING ILLUSTRATES TYPICAL EMPLOYMENT OF NVIATO/VIATO:

```
.  
.  PROGRAM MAIN  
.  EXTERNAL A.A1  
.  CALL NVIATO(  A.A1)  
. 100 CALL VIATO  
.  GO TO 100  
.  END  
  
.  SUBROUTINE A(NAMTO)  
.  CALL NAMTO(<ARGUMENT LIST>)  
.  RETURN  
.  END  
  
.  SUBROUTINE A1(<ARGUMENT LIST>)  
.  <USER CODE>  
.  CALL NVIATO(  B.B1)  
.  RETURN  
.  END  
  
.  SUBROUTINE A2(<ARGUMENT LIST>)  
.  <USER CODE>  
.  CALL EXIT  
.  END  
  
.  SUBROUTINE B(NAMTO)  
.  CALL NAMTO(<ARGUMENT LIST>)  
.  RETURN  
.  END  
  
.  SUBROUTINE B1(<ARGUMENT LIST>)  
.  <USER CODE>  
.  CALL NVIATO(  A.A2)  
.  RETURN  
.  END
```

. NOTE THAT CALLS BETWEEN SUBROUTINES CALLED BY A AND SUBROUTINES CALLED  
. BY B ARE POSSIBLE, EVEN THOUGH THE SUBROUTINES INVOLVED ARE IN DIFFERENT  
. SEGMENTS WHICH OVERLAY THE SAME AREA IN CORE. THE ORDER IN WHICH  
. THE 'TO' SUBROUTINES ARE CALLED IN THE ABOVE EXAMPLE IS AS FOLLOWS:

```
.  A1  
.  B1  
.  A2
```

END

**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**OPENPR  
001**

```

SUBROUTINE OPENPR  & OPEN ALTERNATE PRINT FILE(S)
-----
C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      06/01/75      ORIGINAL CODE
C      MARY TOMPKINS      LEMSCO  01/15/80      USE NEW CHARACTER ROUTINES
C
C
C METHOD
C -----
C
C      IF NO ALTERNATE PRINT FILES SPECIFIED OR IN DATA/CHECKOUT
C      MODE RETURN.
C
C      IF ALTERNATE PRINT FILE EXIST ON TAPE ACKNOWLEDGE WITH NOTE
C      ELSE ASSION ALTERNATE PRINT FILE. PRINT 0, 1, 2, PAGE HEADERS.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE
C
C
C EXTERNAL REFERENCES
C -----
C
C      ERFACL      & RETRIEVE FACILITIES ASSIONMENT INFOMATION
C      MDLOG      & LOG DIAONOSTIC MESSAGES
C      ERCSF      & SUBMIT EXEC-8 CONTROL STATEMENT FUNCTION
C      PRINC      & SET PRINT LINES PER INCH
C      PRTHRO     & SET PRINT MARGINS AND LINES/PAGE ON SPECIFIED UNIT
C      PRCHR      & PRINT BOX CHARACTERS ON SPECIFIED UNIT
C      PRNUM      & PRINT BOX NUMBERS ON SPECIFIED UNIT
C      CST4IN     & CHARACTER STRING FOR INTEGER
C      OETICE     & GET INTEGER-CHARACTER-EQUIVALENT FROM CHARACTER STRING
C
C
C EXCEPTIONS
C -----
C
C      NONE.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
C      INCLUDE KOMALT.LIST      & ALTERNATE PRINT FILE INFO
C
C
C LOCAL DECLARATIONS
C -----

```

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

OPENPR  
002

```

C      INTEGER IDFILE(9)      8 PKT USED TO OBTAIN FILE STATUS
      DATA (IDFILE(N),N=1,2)/*10'. ' /
      INTEGER NODECK(2)/* ' NO DECK *' /

C      INTEGER
C CHAR      000000000111111111222222222233333333334444444444
C      1234567890123456789012345678901234567890123456789
      8 JASOC(8)/*BASO.CP *DAMPRT-N(>1)..F/1/POS/32 . ' /      8 32 POSITIONS
      8 JUSE(8)/*BUSE IN.*DAMPRT-N(>1). . ' /

C
C
C PROCEDURE
C -----
C
      CALL TRACE

C
C
C GET FACILITIES ASSIGNMENT INFO
C
      IF(MALTH.LE.0) GO TO 900      8 NO ALTERNATE PRINT FILES
      IF(MDATAC.NE.0) GO TO 900      8 DATA/CHECKOUT MODE
      CALL ERFACL(IDFILE)
      CALL GETICE(IEQUIP. IDFILE(7),1)

C
C
C IF TAPE IS ALREADY ASSIGNED TO UNIT 10. THEN USE IT FOR ALTERNATE PRINT FILE
C
      IF(IEQUIP.LT.1) GO TO 200      8 NOT ASSIGNED
      IF(IEQUIP.GT.15) GO TO 200      8 NOT TAPE
      NCOPY=0
      MALTH=1
      N=0
      CALL MNOTE(
      *DISPLAY/MAP PRINT FILE WRITTEN ON TAPE UNIT 10.*)
      CALL FILHDO
      RETURN

C
C
C ASSION NEW ALTERNATE PRINT FILE(S) ON DISK
C
200 IF(MBATCH.EQ.0) GO TO 220
      NODECK(1)=.
      NODECK(2)=.
220 CONTINUE
      NMIN=0
      NMAX=MALTH-1
      NITSHI=25*MALTH-1
      DO 500 N=NMIN,NMAX
          CALL CST4IN(JASOC(1),17,1. N,1)
          CALL ERCSF(NAO,JASOC(1)
          CALL CST4IN(JUSE(1),7,1. N,1)
          CALL CST4IN(JUSE(1),17,1. N,1)
          CALL ERCSF(NAO,JUSE(1)
          CALL FILHDO
500 CONTINUE

```

OPENPR  
003

**N-179**



**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**OPENPR  
884**

```

155      FORMAT(' '//
1        8X.'          '.42X.'    *** FILE ***   ')
        CALL PRNUM(NUNIT,12,12.
2        MMIN,BLANK,BLANK,BLANK,BLANK,BLANK,NMAX)
        WRITE(NUNIT,150)
160 CONTINUE
        IF(KPAGE.LT.80) GO TO 180
        CALL PRNUM(NUNIT,16,16.
2        BLANK,BLANK,BLANK,N,BLANK,BLANK,BLANK)
        WRITE(NUNIT,155)
180 CONTINUE

```

C  
C  
C  
C

WRITE 2ND PAGE OF FILE HEADING

```

        WRITE(NUNIT,130)          8PAGE EJECT
        WRITE(NUNIT,115) JRPIDT,N.NODECK
        WRITE(NUNIT,125) (JMDG(1,1),1-1,12)
        ITEMPC=NCOPY
        IF(ITEMP.EQ.0) ITEMPC=-999999      8 NCOPY UNKNOWN
        WRITE(NUNIT,105) ITEMPC,HALTH
105 FORMAT('0'/'0'/'0'/'0'/'
1 10X.8('??????')/'0'/'
2 12X.'DO YOU HAVE:'/' //
3 12X.'      1 COPY OF BASIC PRINT FILE FOR THIS RUN???'//
4 12X. 16.' COPIES OF ALL '.12.
5 ' DISPLAY/MAP PRINT FILES FOR THIS PROGRAM EXECUTION???'/'0'/'
6 10X.8('??????')/'0'/'0')
        RETURN
        END

```

```

SUBROUTINE OPEN3 & OPEN INPUT NSS/RBV FILE ASSIGNED TO UNIT 3
-----
C
C
C
C HISTORY
C -----
C
C
C      J C CRISP      LEC      08/13/79      REQUIREMENTS
C      J C CRISP      LEC      08/18/79      ALGORITHM DESIGN
C      J C CRISP      LEC      08/20/79      ALGORITHM CODING
C      J C CRISP      LEMSCO 08/20/79      IF REC FMT IS 'PXB' DON'T CALL PITROL
C
C
C METHOD
C -----
C
C      INITIALIZE FLAGS IN KOMLU3 AND KOMNER TO WORST CASE (ZERO OR NULL).
C      CHECK DATA/CHECKOUT MODE - IF SET CALL OCONST AND PITROL. SET FILE
C      NAME AND CALL FLINFO. CHECK LU3FID(FIDBFH) AND LU3FID(FIDEQT).
C      IF 'TAPE' OR 'DISK' CALL OP3XXX CORRESPONDING TO EQUIPMENT TYPE.
C      IF EQUIPMENT TYPE = 'ERR', 'OTHER', OR 'NUL' ISSUE APPROPRIATE
C      FATAL MESSAGE. CALL OCONST AND PITROL.
C
C
C MACHINE DEPENDENT CODE
C -----
C
C      NONE
C
C EXTERNAL REFERENCES
C -----
C
C      FLINFO      & GET FACILITIES ASSIGNMENT INFORMATION
C      MOPATL      & PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C      OP3TAP      & DETERMINE IF TAPE IS BIP OR MOP TAPE
C      OP3DSK      & READ HEADER AND ANNOTATION RECORDS FROM DISK
C      OCONST      & DEFINE GEOMETRY OF INPUT DATA
C      PITROL      & ESTIMATE PITCH AND ROLL
C
C
C EXCEPTIONS
C -----
C
C      1. THE FOLLOWING CONDITIONS GENERATE FATAL ERRORS:
C
C          LU3FID(FIDEQT) = 'ERR' OR 'OTHER'
C          LU3FID(FIDBFH) <> ('BST' OR 'BB')
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMNYQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES.COUNTERS
C      INCLUDE FIDEQT.LIST      & DEFINES STRUCTURE OF LU3FID ARRAY
C      INCLUDE KOMLU3.LIST      & PACKET POINTERS FOR UNIT3
C      INCLUDE KOMTOL.LIST      & COMMON BLOCKS AND DEFINE PROCEDURES

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

OPEN3  
002

```

      INCLUDE KONNER.LIST      0 ENTS SCENE PARAMETERS
C
C
C LOCAL DECLARATIONS
C -----
C
C      NONE
C
C
C PROCEDURE
C -----
C
C      CALL TRACE
C
C
C INITIALIZE TO WORST CASE
C
      NERSAT(1) = . . .
      NERSAT(2) = . . .
      NERSEN = . . .
      DO 100 I = 1,3
100  NERTS(1) = 0
      NERLIN = 0
      NERSAM = 0
      NERDAY = 0
      NERMON = . . .
      NERYR = 0
      NERSEL = 0
      NERSAZ = 0
      NERCHA = 0
      ALTKH = 0
      ALTSAM = 0
      CTRLIN = 0
      CTRSAM = 0
      CTRLAT = 0
      CTRLON = 0
      DINLAT = 0
      PITDEO = 0
      ROLDEO = 0
      YANDEO = 0
      WNYDEO = 0
      NCCT = 0
      NCCTOT = 0
      NERGE0 = 'BAD'
      NERRS = . . .
      NERCOR = . . .
      DO 150 I = 1,10
      NERBAN(1) = . . .
      NERBAI(1) = . . .
      NERTHO(1) = . . .
150  CONTINUE
      NERADN = . . .
      NERPAT = . . .
      NERRON = . . .
      LUTLOF = 0
      LUTBIL = 0

```

**DAN PACKAGE APPENDIX H  
UTILITY ROUTINES**

**OPEN3  
003**

```

    LUSNID = 0
    LUSCOR = 0
    LUSMOR = 0
    LUSBID = 0
    LUSDFN = 'NUL'
    LUSSEQ(1) = 'NUL'
    LUSSEQ(2) = '0'
    LUSREF(1) = 'NUL'
    LUSREF(2) = '0'
    DO 100 I = 1,VSMAX
        LUSREL(1) = ' '
        LUSRLO(1) = 0
        LUSRHI(1) = 0
100 CONTINUE
    LUSVOL = 0
    LUSVHI = 0
    LUSRDF = 0

C
C
C CHECK IF 'N' DATA CHECKOUT MODE
C
    IF(INDATAC.NE.0)NERTS(1) = 1
    IF(NERTS(1).EQ.1) GO TO 900

C
C
C SET FILE NAME AND GET FACILITIES ASSIGNMENT INFORMATION
C
    LUSPKT(1) = '3'
    LUSPKT(2) = ' '

C
    CALL FLINFO(LUSFID, LUSPKT, '00')

C
C
C CHECK EQUIPMT. TYPE AND CALL APPROPRIATE ROUTINE
C
200 LUSDFN = LUSFID(FIDDFN)
    IF(LUSFID(FIDEQT).EQ.'TAPE') CALL OP3TAP
    IF(LUSFID(FIDEQT).EQ.'DISK') CALL OP3DSK
    IF(LUSFID(FIDEQT).NE.'TAPE'.AND.LUSFID(FIDEQT).NE.'DISK')
        & CALL M0FATL('INPUT TAPE/DISK NOT ASSIGNED TO UNIT 3')

C
C
C SET GEOMETRY AND PITCH AND ROLL CONSTANTS
C
300 CALL OCONST
    IF(LUSREF(1).NE.'PK0') CALL PITROL(
        & CTRLAT,CTRLOM,DIRLAT,DIRLOM,MHYDEG,ALTKM,
        & PITDEG,ROLDEG)

C
C
    RETURN

C
    END

```

SUBROUTINE OPN12N & OPEN INPUT DETECTION FILES

```

C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LE.      12/19/73      ORIGINAL CODE
C      CHARLES HELMKE      LEC      12/13/79      LARGER HDR. VARIABLE BINTYPE & REC SZ
C      J C CRISP           LEMSCO 05/16/80      PUT NAMES IN I/O PKTS. NERCHA=NOUICH
C
C
C METHOD
C -----
C
C      IF MDTAC.NE.0 LOAD REGISTRATION PARAMETERS AND INITIALIZE NOMINAL
C      INPUT WINDOW PKT. ELSE. ATTEMPT TO OPEN ALL POSSIBLE DETECTION
C      FILES. READ HEADER SECTOR FROM EACH PRESENT DETECTION FILE INTO
C      LOCAL BUFFER. FOR EACH FILE:
C          IF FIRST PRESENT FILE (NERTS(1)=0). THEN LOAD KOMNER. KOMKLS.
C          KOMFIT. KOMDET FROM BUFFER.
C          IF NOT FIRST PRESENT FILE (NERTS(1)<>0). THEN COMPARE KOMNER.
C          KOMKLS. KOMFIT. WITH BUFFER CONTENTS. FLAG MISMATCHES.
C          AND LOAD KOMDET VARIABLES WITH CORRESPONDING NON-ZERO.
C          NON-BLANK LOCATIONS IN BUFFER.
C      CHECK CONTINUITY OF LINES AND SAMPLES BETWEEN DETECTION FILES
C      COMPUTE CLASSIFICATION ENVELOPE FLAG/MARK DETECTION FILES
C      DEFECTIVE OR NOT PRESENT.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE
C
C EXTERNAL REFERENCES
C -----
C
C      ERCSF      & SUBMIT EXEC-8 CONTROL STATEMENT FUNCTION
C      MDNOTE      & PRINT/COUNT/LOG 'NOTE' DIAGNOSTIC MESSAGE
C      MDFATL      & PRINT/COUNT/LOG 'FATAL ERROR' DIAGNOSTIC MESSAGE
C      ER10W      & INITIATE I/O & WAIT FOR COMPLETION
C      LOREQ8      & LOAD NOMINAL REGISTRATION PARAMETERS
C      CST4IN      & CHAR STRING FOR INTEGER
C
C
C EXCEPTIONS
C -----
C
C      1. THE FOLLOWING CONDITION GENERATE THE DIAGNOSTICS SHOWN:
C
C          CONDITION                                DIAGNOSTICS
C      DETECTION FILE NOT AVAILABLE                MDNOTE
C      NO DETECTION FILES                          MDFATL
C      DETECTION FILE IN USE                        MDFATL

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

OPN12N  
002

```

C      DEFECTIVE DETECTION FILE                                MOFATL
C      INCONSISTENCIES BETWEEN DETECTION
C      FILE HEADERS                                           MOFATL
C      DISCONTINUITY OF LINES BETWEEN
C      DETECTION FILES                                         MOFATL
C      DISCONTINUITY OF SAMPLES BETWEEN
C      DETECTION FILES                                         MOFATL
C
C
C      GLOBAL DECLARATIONS
C      -----
C
C      INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES. COUNTERS
C      INCLUDE KOHL2N.LIST      & COMMON I/O PACKETS FOR DETECTION FILES (21-24)
C      INCLUDE KOMIO.LIST       & COMMON I/O FUNCTIONS
C      INCLUDE KOMNER.LIST      & COMMON ERTS SCENE PARAMETERS
C      INCLUDE KOMKLS.LIST      & COMMON CLASSIFICATION INFO
C      INCLUDE KOMFIT.LIST      & COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C      INCLUDE KOHDET.LIST      & COMMON DETECTION FILE WINDOWS
C      INCLUDE MINDEF.LIST      & DEFINE STRUCTURE OF WINDOW PACKETS
C      INCLUDE KOMINH.LIST      & COMMON INPUT WINDOW PACKETS
C      INCLUDE KOMOHV.LIST      & COMMON OUTPUT WINDOW PACKETS
C      INCLUDE TRFORM.LIST      & DEFINE COORDINATE TRANSFORMATION FUNCTIONS
C      INCLUDE FACBIT.LIST      & EXEC-8 FACILITY REQUEST STATUS BIT MNEMONICS
C
C
C      LOCAL DECLARATIONS
C      -----
C
C      INTEGER KBUFR(20,9)      & INTERNAL WORK BUFFER FOR DETECTION FILES
C      INTEGER IRONER           & ERROR COUNTER FOR KOMNER
C      INTEGER IROKLS           & ERROR COUNTER FOR KOMKLS
C      INTEGER IROFIT           & ERROR COUNTER FOR KOMFIT
C      INTEGER IFIRST           & FIRST AVAILABLE STRIP
C      INTEGER ILAST            & LAST AVAILABLE STRIP
C      INTEGER NLAST            & STRIP BEFORE LAST STRIP (LAST-1)
C      INTEGER
C
C      CHAR      00000(1001111111111222222222233333333334
C      12345678901234567890123456789012345678901234567890
C      * JASO2(4) /* SASO.AX *DAMDET-N. . . . .
C      * JUSE2(4) /* SUSE 2N.*DAMDET-N. . . . .
C      * JFREED(4) /* FREE.D      2N. . . . .
C      * NOFILE(4) /* NO FILE *DAMDET-N.'.-0/
C      * JFLBAD(8) /* FILE *DAMDET-N DEFECTIVE. .'-0/
C
C
C      PROCEDURE
C      -----
C
C      CALL TRACE('OPN12N')
C
C
C      INITIALIZE FOR DATA/CHECKOUT MODE
C
C      IF(MDATAC.EQ.0) 00 TO 300      & NOT DATA/CHECKOUT MODE
C      CALL LDORON      & LOAD REGISTRATION PARAMETERS GENERATED BY CONTROL

```

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

OPN12H  
003

```

      HSAIHW(WLIN,WMIN)=-100
      HSAIHW(WLIN,WMAX)=NERLIN+100
      HSAIHW(WSAM,WMIN)=-100
      HSAIHW(WSAM,WMAX)=NERSAN+100
      GO TO 900

C
C
C INITIALIZE FOR NORMAL MODE
C
      300 NERTS(1)=0
          IRONER=0
          IROKLS=0
          IROFIT=0
          HSAIHW(WLIN,WMIN)=+99999999
          HSAIHW(WLIN,WMAX)=-99999999
          HSAIHW(WSAM,WMIN)=+99999999
          HSAIHW(WSAM,WMAX)=-99999999

C
C
C ATTEMPT TO OPEN ALL 4 POSSIBLE DETECTION FILES
C
      DO 800 JCCT=1,4

C
C
C PUT LOGICAL UNIT NAMES (20*JCCT) IN DETECTION FILE I/O PACKETS
C
      CALL MOVCS( L2NPKT(1,JCCT),(1),(12),
-              '2',(1),(1), ' ' )
      CALL PUTICE( L2NPKT(1,JCCT),(2),
-              ICE('0')+JCCT)

C
C
C ASSION FILE
C
      CALL CST4IN(JAS0A2,17.1, JCCT,1)
      CALL ERCSF(NA0, JAS0A2)
      IF(NOTCAT(NA0).EQ.0) GO TO 330
      CALL CST4IN(NOFILE,17.1, JCCT,1)
      CALL MDNOTE( NOFILE)
      JENMOY(JCCT)= ' '
      GO TO 590
330 IF(OTHRUN(NA0).EQ.0) GO TO 350
      CALL MOFATL(
-      'DETECTION FILE IN USE BY ANOTHER RUN WITH SAME RUNID/QUAL ' )
      JENMOY(JCCT)= ' '
      GO TO 590
350 CALL CST4IN(JUSE2,7.1, JCCT,1)
      CALL CST4IN(JUSE2,17.1, JCCT,1)
      CALL ERCSF(NA0, JUSE2)

C
C
C READ HEADER (SECTORS 0 THRU 8) INTO BUFFER
C
      CALL RMDR(KBUFR, L2NPKT(1,JCCT))
      IF(KBUFR(4,1).EQ.0) GO TO 580      & NERTS(1)=0 -- DETECTION FILE IN ERROR
      IF(KBUFR(63,1).NE.JCCT) GO TO 580      & *DAMDET-<N> NOT FOR CCT = <N>

```

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

OPN12N  
004

```

      IF(NERTS(1).NE.0) GO TO 440      & NOT FIRST DETECTION FILE
C
C
C FIRST FILE -- LOAD SECTORS 0 THRU 6 INTO KONNER, KOMKLS, AND KONFIT
C
      DO 411 NWD=1,SIZNER
411 KONNER(NWD)=KBUFR(NWD,1)
      DO 412 NWD=1,SIZKLS
412 KOMKLS(NWD)=KBUFR(NWD,4)
      DO 413 NWD=1,SIZFIT
413 KONFIT(NWD)=KBUFR(NWD,6)
      GO TO 450
C
C
C SUBSEQUENT FILE -- COMPARE SECTORS 0 THRU 6 WITH KONNER, KOMKLS, AND KONFIT
C
440 DO 441 NWD=1,62      & EXCLUDE NCCT & NCCTOT
      IF(KONNER(NWD).NE.KBUFR(NWD,1)) IRONER=IRONER+1
441 CONTINUE
      DO 442 NWD=1,20      & EXCLUDE KTIPIX
      IF(KOMKLS(NWD).NE.KBUFR(NWD,4)) IROKLS=IROKLS+1
442 CONTINUE
      DO 443 NWD=1,17
      IF(KONFIT(NWD).NE.KBUFR(NWD,6)) IROFIT=IROFIT+1
443 CONTINUE
C
C
C ALL FILES -- LOAD NULL WORDS IN KONDET FROM SECTORS 7 THRU 8
C
450 DO 460 NWD=1,SIZDET
      IF((KONDET(NWD).EQ.0).OR.
      & (KONDET(NWD).EQ.' ')) KONDET(NWD)=KBUFR(NWD,8)
460 CONTINUE
C
C
C COMPUTE CLASSIFICATION ENVELOPE
C
      MSA1WW(WLIN,WMIN)=MIN0(MSA1WW(WLIN,WMIN),MSADWW(WLIN,WMIN,JCCT))
      MSA1WW(WLIN,WMAX)=MAX0(MSA1WW(WLIN,WMAX),MSADWW(WLIN,WMAX,JCCT))
      MSA1WW(WSAM,WMIN)=MIN0(MSA1WW(WSAM,WMIN),MSADWW(WSAM,WMIN,JCCT))
      MSA1WW(WSAM,WMAX)=MAX0(MSA1WW(WSAM,WMAX),MSADWW(WSAM,WMAX,JCCT))
C
      IF(MERGE0.NE.'ERT') GO TO 470
C
      COR1WW(WLIN,WMIN)=CORL4A(MSA1WW(WLIN,WMIN),MSA1WW(WSAM,WMIN))
      COR1WW(WLIN,WMAX)=CORL4A(MSA1WW(WLIN,WMAX),MSA1WW(WSAM,WMAX))
      COR1WW(WSAM,WMIN)=CORL4A(MSA1WW(WLIN,WMIN),MSA1WW(WSAM,WMIN))
      COR1WW(WSAM,WMAX)=CORL4A(MSA1WW(WLIN,WMAX),MSA1WW(WSAM,WMAX))
      GO TO 600
470 COR1WW(WLIN,WMIN)=MSA1WW(WLIN,WMIN)
      COR1WW(WLIN,WMAX)=MSA1WW(WLIN,WMAX)
      COR1WW(WSAM,WMIN)=MSA1WW(WSAM,WMIN)
      COR1WW(WSAM,WMAX)=MSA1WW(WSAM,WMAX)
      GO TO 600
C
C

```



**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**OPN12N  
005**

**C FLAG DEFECTIVE DETECTION FILE**

```
C
C
500 CALL CST4IN(JFLBAD,17,1, JCCT,1)
    CALL MOFATL( JFLRAC)
    JENHDY(JCCT)=' ERROR'
```

**C MARK DETECTION FILE DATA NOT PRESENT**

```
C
590 MSADWH(WLIN,WMIN,JCCT)=0
    MSADWH(WLIN,WMAX,JCCT)=0
    MSADWH(WSAM,WMIN,JCCT)=0
    MSADWH(WSAM,WMAX,JCCT)=0
```

**C 600 CONTINUE 3 LOOP TO ASSIGN NEXT DETECTION FILE**

```
C
C
C CHECK DETECTION FILES FOR INTER-FILE COMPATIBILITY.
C FILES MUST DEFINE AN AREA SUCH THAT FOR EACH SCAN LINE
C THERE ARE CONTIGUOUS SAMPLES FROM THE FIRST SAMPLE TO
C LAST SAMPLE. THIS RESTRICTION IS NECESSARY BECAUSE
C THE BUFFER STRUCTURE (PXBOEF) FOR LINES IMPLIES THAT
C THERE ARE NO GAPS FROM PXLSAM TO PXHSAM.
```

**C FIND FIRST AND LAST DEFINED STRIPS**

```
C
    IFIRST = 0
    LAST = 0
    DO 610 I = 1,4
        IF(MSADWH(WLIN,WMIN,I).EQ.0) GO TO 610
        IF(IFIRST.EQ.0) IFIRST = I
        LAST = I
```

```
610 CONTINUE
    IF(IFIRST.EQ.0) GO TO 640
```

```
C
640 IF( (LAST-IFIRST+1).LT.3) GO TO 700 3 1 STRIP OR 2 ADJACENT
    ISTATE = 0
    NLAST = LAST - 1
```

```
C
    DO 670 I = IFIRST,NLAST
        IF(MSADWH(WLIN,WMIN,I).EQ.MSADWH(WLIN,WMIN,I+1)) GO TO 670
        IF(MSADWH(WLIN,WMIN,I).GT.MSADWH(WLIN,WMIN,I+1)) GO TO 650
        ISTATE = -1
        GO TO 670
```

```
650 IF(ISTATE.EQ.-1) GO TO 800
    ISTATE = 1
```

```
670 CONTINUE
    ISTATE = 0
```

```
C
    DO 690 I = IFIRST,NLAST
        IF(MSADWH(WLIN,WMAX,I).EQ.MSADWH(WLIN,WMAX,I+1)) GO TO 690
        IF(MSADWH(WLIN,WMAX,I).EQ.MSADWH(WLIN,WMAX,I+1)) GO TO 680
        ISTATE = 1
        GO TO 690
680 IF(ISTATE.EQ.1) GO TO 810
```

OPN12N  
006

**N-109**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

OPN12N  
007

```
      CALL CST4IN(JFREED,17.1, JCCT,1)
      WRITE(6,725) JCCT
725  FORMAT(' (FILE 'DANDET-'.J1.' DELETED)')
      CALL ERCSF(NAO, JFREED)
700  CONTINUE
      RETURN
```

C  
C  
C  
C  
C

```
SUBROUTINE R0HDR(KBUFR, NPKT)
DIMENSION NPKT(1)
IOSIZE(NPKT)=252      & HEADER SIZE IN WORDS FOR OPN12N
IOADDR(NPKT)=LOC(KBUFR)
IOSECT(NPKT)=0
IOFUNC(NPKT)='&K'    & READ HEADER
CALL ER10H(NPKT)
RETURN
END
```

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

OP3BIP  
001

SUBROUTINE OP3BIP 3 OPEN INPUT ERTS MSS TAPE

```

C
C
C
C HISTORY
C -----
C
C   CHARLES MELNKE      LEC    08/07/79    REQUIREMENTS
C   M A TOMPKINS       LEC    10/02/79    ALGORITHM DESIGN
C   M A TOMPKINS       LEC    10/11/79    ALGORITHM CODING
C
C
C METHOD
C -----
C
C   KTBLTY:  * * *.READ ID RECORD INTO KTABLE AND EXTRACT BINARY
C             INFORMATION. READ ANNOTATION RECORD INTO KTABLE AND CONVERT
C             EBCDIC ANNOTATION RECORD TO CHARACTER STRING AND EXTRACT/DECODE
C             CHARACTER INFORMATION. VERIFY AND COMPARE INFORMATION FROM ID
C             AND ANNOTATION RECORDS.SET: LU3SEQ.LU3REF.LU3WIB.LU3BIB.
C             INITIALIZE INPUT WINDOW PACKET.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C   NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C   MDFATL      3 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C   ERPRNT      3 PRINT IMAGES ON TTY OR LINE PRINTER (CHARACTER)
C   GETBYT      3 GET ONE NON-NEG INTEGER FROM BYTE STRING
C   GETDBY      3 GET NON-NEG INTEGER FROM DOUBLE BYTE FROM BYTE STRING
C   CST4EB      3 CHARACTER STRING FOR EBCDIC BYTE STRING
C   HDWARN      3 PRINT/LOG/COUNT 'WARNING' MESSAGES
C   MOVCHR      3 MOVE A CHARACTER FROM ONE STRING TO ANOTHER
C   DCODE       3 DECODE NUMERIC CHARACTER STRING
C   GETCHR      3 GET CHARACTER FROM CHARACTER STRING
C   MOVCS1      3 MOVE SUBSTRING 2 TO SUBSTRING 1
C   ER10W       3 INITIATE I/O AND WAIT FOR COMPLETION
C   BST4BB      3 INTERNAL BYTE STRING FOR 8-BIT EXTERNAL BYTE STRING
C   HDNOTE      3 PRINT/LOG 'NOTE' MESSAGES
C   CST4IN      3 CHARACTER STRING FOR INTEGER
C   DOUBLE PRECISION CDS4CS 3 VARIABLE LENGTH CST FOR FIXED LENGTH CST
C   INTEGER NC4NI 3 # OF CHARS FOR # OF INTEGERS
C   INTEGER NI4NC 3 # OF INTEGERS FOR # OF CHARS
C   INTEGER LCHREQ 3 LOCATION OF CHAR IN STRING - SEARCH CHAR
C   INTEGER NB4NI 3 # OF BYTES FOR # OF INTEGERS
C   REAL         DEG 3 DEGREES FOR DEG.MIN.SEC
C
C
C EXCEPTIONS
C -----

```

OP301P  
002

**N-192**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

OP301P  
003

```

C -----
C
C      CALL TRACE
C
C
C INITIALIZE REEL IDENTIFIERS
C
C      LU3REL(1) = LU3FID(FIDCRL)  & CURRENT (AND ONLY) REEL
C      LU3VOL = 1
C      LU3VHI = 1
C
C
C INITIALIZE TO '0' VERSIONS
C
C      KTBITY = ' '  & FLAG PREVIOUS CONTENTS KTABLE AS DESTROYED
C      LU3SEQ(1) = '01P'
C      LU3SEQ(2) = '0'  & INITIALIZE TO 4 CHANNEL VERSION OF DATA SEQ
C      NERCHA = 4
C      LU3REF(1) = 'ERT'
C      LU3REF(2) = '0'  & INITIALIZE TO ERTS(72) VERSION OF REC FMT
C      NDSAVE = NDTOTL
C
C
C READ ID RECORD INTO KTABLE.
C
C      IOADDR(LU3PKT) = LOC(KTABLE)
C      IOSIZE(LU3PKT) = 10
C      IOWAIT(LU3PKT) = 0
C      IOFUNC(LU3PKT) = 'BK'  & READ FORWARD
C      CALL ERION(LU3PKT)
C      ISTAT = ICODE(LU3PKT)
C      IF( (ISTAT.EQ.'EOF').OR.(ISTAT.EQ.'BADF') )CALL NDFATL(
C      = CBS4CS(ISTAT,1,4),' WHILE READING ID RECORD')
C      IF( (ISTAT.EQ.'LOST').OR.(ISTAT.EQ.'BADR') )CALL MONOTE(
C      = CBS4CS(ISTAT,1,4),' WHILE READING ID RECORD')
C
C
C IF ID RECORD IS IN BB FORMAT. THEN CONVERT IT TO BST
C
C      IF(LU3BFH.EQ.'BB')CALL BST4BB(KTABLE,  KTABLE,NB4NI(38))
C
C
C EXTRACT BINARY INFORMATION FROM ID RECORD
C
C      CALL GETBYT(NBYTOR,  KTABLE,17)  & DATA RECORD LENGTH IN BYTES
C      CALL GETBYT(NBERTS(1),  KTABLE,19)  & ERTS PROJECT NUMBER
C      CALL GETBYT(NBERTS(2),  KTABLE,20)  & DAYS SINCE LAUNCH
C      NTERTS = NBERTS(2)*2**6
C      CALL GETBYT(NBERTS(2),  KTABLE,21)
C      NBERTS(2) = NBERTS(2) + NTERTS
C      CALL GETBYT(NBERTS(3),  KTABLE,22)  & GMT
C      NTERTS = NBERTS(3)*1000
C      CALL GETBYT(NBERTS(3),  KTABLE,23)
C      NTERTS = NTERTS + NBERTS(3)*10
C      CALL GETBYT(NBERTS(3),  KTABLE,24)
C      NBERTS(3) = NBERTS(3) + NTERTS

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

OP3BIP  
004

```

        CALL GETBY(INBYTAL, KTABLE,30) & ADJUSTED LINE LEN IN BYTES
        CALL GETBY(INODTHP, KTABLE,30) & MODE/CORRECTION
C
C
C CONVERT EBCDIC ID RECORD TO CHARACTER STRING
C
        CALL CST4EB(KTABLE, KTABLE,30)
C
C
C DETERMINE VERSION OF ID RECORD
C
        CALL GETCHR(IDASH, KTABLE,5)
        NPST70 = 0
        IF(IDASH.NE.'-') NPST70 = 1
        IF(IDASH.NE.'-') LU3REF(2) = '2' & ERTS(JAN70)
C
C
C EXTRACT CHARACTER INFORMATION FROM ID RECORD
C
C
        CALL DCODE(NERTS(1),REAL,KODTYP, KTABLE,1,1) & ERTS PROJECT NO.
        IF(KODTYP.NE.'IN')NERTS(1) = MAXINT
C
        CALL DCODE(NERTS(2),REAL,KODTYP, KTABLE,2,3+NPST70) & DAYS/LAUNCH
        IF(KODTYP.NE.'IN')NERTS(2) = MAXINT
C
        CALL DCODE(NERTS(3),REAL,KODTYP, KTABLE,6+NPST70,5) & OMT
        IF(KODTYP.NE.'IN')NERTS(3) = MAXINT
C
        CALL DCODE(NCCT,REAL,KODTYP, KTABLE,14,1) & TAPE SEQ NCCT OF NCCTOT
        IF(KODTYP.NE.'IN')NCCT = MAXINT
C
        CALL DCODE(NCCTOT,REAL,KODTYP, KTABLE,16,1) & TAPE SEQ NCCT OF NCCTOT
        IF(KODTYP.NE.'IN')NCCTOT = MAXINT
C
C
C PRINT REC IF TRACE IS ON OR DIAGNOSTIC COUNT IS CHANGED
C
        IF( (NTRACE.EQ.0).AND.(NOSAVE.EQ.NDTOTL) ) GO TO 150
        CALL ERPRNT(1,N14NC(40), KTABLE)
        NOSAVE = NDTOTL
150 CONTINUE
C
C
C READ ANNOTATION RECORD INTO KTABLE
C
        IOADDR(LU3PKT) = LOC(KTABLE)
        IOSIZE(LU3PKT) = 30
        IOHAIT(LU3PKT) = 0
        IOFUNC(LU3PKT) = '8K' & READ FORWARD
        CALL ERION(LU3PKT)
        ISTAT = ICODE(LU3PKT)
        IF( (ISTAT.EQ.'EOF').OR.(ISTAT.EQ.'BADF') )CALL MDEFATL(
        = COS4CS(ISTAT,1,4),' WHILE READING ANNOTATION RECORD')
C
        IF( (ISTAT.EQ.'BADR').OR.(ISTAT.EQ.'LOST') ) CALL MNOTE(

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

OPSBIP  
005

```

      - CDS4CS(ISTAT.1.4),' WHILE READING ANNOTATION RECORD')
C
C
C IF ANNOTATION RECORD IS IN 88 FORMAT. THEN CONVERT IT TO 88T
C
      IF(LUZ8FH.EQ.'88') CALL 88T488(KTABLE, KTABLE.NB4NI(38))
C
C
C CONVERT EBCDIC ANNOTATION RECORD TO CHARACTER STRING
C
      CALL C8T4EB(KTABLE, KTABLE.144)
C
C
C EXTRACT CHARACTER INFORMATION FROM ANNOTATION RECORD
C
      CALL DCODE(NERDAY.REAL,KODTYP, KTABLE.1.2) 8 DAY EXPOSURE
      IF(KODTYP.NE.'IN') NERDAY = 99
      IF(NERDAY.LT.1.(R.NERDAY.0T.3))CALL MDHARN(
      - 'BAD EXPOSURE DAY FROM ANNOTATION RECORD')
C
      CALL MOVCS(NERMON.1.3, KTABLE.3.3,' ') 8 MONTH
C
      CALL DCODE(NERYR.REAL,KODTYP, KTABLE.8.2) 8 YEAR
      IF(KODTYP.NE.'IN')CALL MDHARN(
      - 'BAD EXPOSURE YEAR FROM ANNOTATION RECORD')
C
      CALL DCODE(LATD.REAL,KODTYP, KTABLE.12.2) 8 SCENE CENTER LAT-DEG
      IF(KODTYP.NE.'IN')CALL MDFATL(
      - 'BAD SCENE CENTER DEG LAT FROM ANNOTATION RECORD')
C
      CALL DCODE(LATH.REAL,KODTYP, KTABLE.15.2) 8 SCENE CENTER LAT-MIN
      IF(KODTYP.NE.'IN')CALL MDHARN(
      - 'BAD SCENE CENTER MIN LAT FROM ANNOTATION RECORD -- 30 ASSUMED')
      IF(KODTYP.NE.'IN')LATH = 30
C
      CALL DCODE(LOND.REAL,KODTYP, KTABLE.18.3) 8 SCENE CENTER LON-DEG
      IF(KODTYP.NE.'IN')CALL MDFATL(
      - 'BAD SCENE CENTER DEG LON FROM ANNOTATION RECORD')
C
      CALL DCODE(LONM.REAL,KODTYP, KTABLE.23.2) 8 SCENE CENTER LON-MIN
      IF(KODTYP.NE.'IN')CALL MDHARN(
      - 'BAD SCENE CENTER MIN LON FROM ANNOTATION RECORD -- 30 ASSUMED')
      IF(KODTYP.NE.'IN')LONM = 30
C
C
C
C DETERMINE VERSION OF ANNOTATION RECORD
C
      CALL GETCHR(1BLANK, KTABLE.34)
      IF(1BLANK.EQ.' 'AND.LUZREF(2).EQ.'0')LUZREF(2) = '1' 8 ERTS(JUL77)
      IF(LUZREF(2).NE.'0') GO TO 200
C
C
C EXTRACT CHARACTER INFORMATION FROM PRE-JUL77 ANNOTATION REC (VERSION '0')
C
      CALL DCODE(1NLATD.REAL,KODTYP, KTABLE.28.2) 8 NADIR LAT-DEG

```



**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**OPSBIP  
000**

```

      IF(KOOTYP.NE.'IN')CALL M0FATL(
      • 'BAD NADIR DEG LAT FROM ANNOTATION RECORD')
C
      CALL DCODE(INLATN.REAL,KOOTYP, KTABLE,32,2) & NADIR LAT-MIN
      IF(KOOTYP.NE.'IN')CALL M0HARN(
      • 'BAD NADIR MIN LAT FROM ANNOTATION RECORD-30 ASSUMED')
      IF(KOOTYP.NE.'IN')INLATN = 30
C
      CALL DCODE(INLOND.REAL,KOOTYP, KTABLE,36,3) & NADIR LON-DEG
      IF(KOOTYP.NE.'IN')CALL M0FATL(
      • 'BAD NADIR DEG LON FROM ANNOTATION RECORD')
C
      CALL DCODE(INLONH.REAL,KOOTYP, KTABLE,40,2) & NADIR LON-MIN
      IF(KOOTYP.NE.'IN')CALL M0HARN(
      • 'BAD NADIR MIN LON FROM ANNOTATION RECORD-30 ASSUMED')
      IF(KOOTYP.NE.'IN')INLONH = 30
C
      CALL DCODE(INERSEL.REAL,KOOTYP, KTABLE,61,2) & SUN ELEVATION
      IF( (INERSEL.LT.0).OR.(INERSEL.GT.90) )KOOTYP = 'ERR'
      IF(KOOTYP.NE.'IN')CALL M0HARN(
      • 'BAD SUN ELEVATION FROM ANNOTATION RECORD')
C
      CALL DCODE(INERSAZ.REAL,KOOTYP, KTABLE,66,3) & SUN AZIMUTH
      IF( (INERSAZ.LT.-360).OR.(INERSAZ.GT.360) )KOOTYP = 'ERR'
      IF(KOOTYP.NE.'IN')CALL M0HARN(
      • 'BAD SUN AZIMUTH FROM ANNOTATION RECORD')
C
      CALL DCODE(INOYAW.REAL,KOOTYP, KTABLE,70,3) & HEADING MINUS YAW (INT)
      IF(KOOTYP.NE.'IN')CALL M0FATL(
      • 'BAD HEADING MINUS YAW FROM ANNOTATION RECORD')
      HMYDEG = INOYAW
C
      MERGED = 'ZRT'
C
C DETERMINE SYSTEM CORRECTION
C
      ISYNCR = M00THP
      NBITIS = 0
      IF(MOD(ISYNCR,2).NE.0) NBITIS = 1
      ISYNCR = ISYNCR / 2
      NBITIS = 0
      IF(MOD(ISYNCR,2).NE.0) NBITIS = 1
      IF( (NBITIS.EQ.0).AND.(NBITIS.EQ.0) ) NERCOR = 'U'
      IF( (NBITIS.EQ.1).AND.(NBITIS.EQ.1) ) NERCOR = 'S'
      IF(NERCOR.EQ.'')CALL M0HARN(
      • 'CALIBRATION AND LINE-LENGTH-ADJUST NOT SAME IN BYTE 30 ID REC
      LORD')
      GO TO 220
C
C
C EXTRACT CHARACTER INFORMATION FROM POST-JUL77 ANNOTATION
C RECORD (VERSION '1' OR '2')
C
      200 CALL DCODE(INERSEL.REAL,KOOTYP, KTABLE,66,2) & SUN ELEVATION
      IF(KOOTYP.NE.'IN')CALL M0HARN(

```

**SAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**OP2BIP  
007**

```

      = 'BAD SUN ELEVATION FROM ANNOTATION RECORD')
C
      CALL DCODE(HERSAZ,REAL,KOOTYP, KTABLE.72.3) & SUN AZIMUTH
      IF(KOOTYP.NE.'IN')CALL MDHARN!
      = 'BAD SUN AZIMUTH FROM ANNOTATION RECORD')
C
      NMYD00 = -9999
      NLAT0 = -9999
      NLATN = 0
      NLON0 = -9999
      NLONN = 0
      CALL MDNOTE('JUL77 0000000 FORMAT -- NO NADIR')
      CALL MDNOTE('ATTITUDE UNDEFINED -- SEC END OF CCT 4')
C
      CALL GETCHR(NERADN, KTABLE.26) & 'A' OR 'D' ORBIT DIRECTION
C
      CALL GETCHR(NERCOR, KTABLE.76) & CORRECTION APPLIED
C
      CALL NOVCST(NERPAT,1.3, KTABLE.27.3.' ') & NOMINAL PATH
C
      CALL NOVCST(NERROW,1.3, KTABLE.31.3.' ') & NOMINAL ROW
C
      CALL GETCHR(10EO, KTABLE.70) & PROJECTION GEOMETRY
C
      LOC0EO = LCHREG(10EO,1.4,10EO) + 1
      NERGEO = 10AEO(LOC0EO)
C
C CONVERT INFORMATION FROM ANNOTATION RECORD
C
      200 CTRLAT = DEG(LAT0,LATN,0.0)
      CTRLON = DEG(LON0,LONN,0.0)
      DIRLAT = DEG(NLAT0,NLATN,0.3)
      DIRLON = DEG(NLON0,NLONN,0.0)
C
C
C VERIFY/COMPARE INFORMATION FROM ID RECORD
C
      NBYTOR = NBYTOR - 56 & SUBTRACT CALIB. INFO
      NERSAM = NBYTAL
      IF(NBYTAL.EQ.NBYTOR) GO TO 300
      IF(MOD(NBYTAL,4).NE.0) GO TO 200
      IF( (NERSAM.LT.2000).OR.(NERSAM.GT.7000) ) GO TO 200
      CALL MDHARN('CONFLICT IN SAMPLES/CCT -- COMPUTED FROM ADJUSTED
      = LINE LENGTH')
      GO TO 300
C
      200 NERSAM = MAXINT
      IF(MOD(NBYTOR,4).EQ.0)NERSAM = NBYTOR
      IF( (NERSAM.LT.2000).OR.(NERSAM.GT.7000) ) CALL MDPATL(
      = 'INVALID SAMPLES/CCT')
      CALL MDHARN(
      = 'CONFLICT IN SAMPLES/CCT -- COMPUTED FROM DATA RECORD LENGTH')
C
C
C VERIFY BEST VALUE OF PROJECT NO.,DAY,TIME RELATIVE TO LAUNCH:
C

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

0P381P  
000

C -- PROJECT NUMBER

C

```
300 IOUT = MAXINT
   BAD = ( (NBERTS(1).LT.1).OR.(NBERTS(1).GT.8) )
   IF(BAD) CALL MDHARN(
   = 'BAD BINARY ERTS NUMBER')
   IF(.NOT.BAD) IOUT = NBERTS(1)
   BAD = ( (NBERTS(1).LT.1).OR.(NBERTS(1).GT.8) )
   IF(BAD) CALL MDHARN('BAD EBCDIC ERTS NUMBER')
   IF(.NOT.BAD) IOUT = NBERTS(1)
   NBERTS(1) = IOUT
```

C

C

C -- DAY RELATIVE TO LAUNCH AT OBSERVATION

C

```
IOUT = MAXINT
   BAD = ( (NBERTS(2).LT.0).OR.(NBERTS(2).GT.9999) )
   IF(BAD) CALL MDHARN(
   = 'BAD BINARY DAYS SINCE LAUNCH')
   IF(.NOT.BAD) IOUT = NBERTS(2)
   BAD = ( (NBERTS(2).LT.1).OR.(NBERTS(2).GT.9999) )
   IF(BAD) CALL MDHARN('BAD EBCDIC DAYS SINCE LAUNCH')
   IF(.NOT.BAD) IOUT = NBERTS(2)
   NBERTS(2) = IOUT
```

C

C

C -- HOUR.MINUTE.TENS OF SECONDS RELATIVE TO OBSERVATION

C

```
IOUT = MAXINT
   BAD = ( (NBERTS(3).LT.0).OR.(NBERTS(3).GT.24000) )
   IF(BAD) CALL MDHARN('BAD BINARY GMT')
   IF(.NOT.BAD) IOUT = NBERTS(3)
   BAD = ( (NBERTS(3).LT.0).OR.(NBERTS(3).GT.24000) )
   IF(BAD) CALL MDHARN('BAD EBCDIC GMT')
   IF(.NOT.BAD) IOUT = NBERTS(3)
   NBERTS(3) = IOUT
```

C

```
DO 310 I = 1,3
   IF(NBERTS(I).EQ.MAXINT) CALL MDFATL(
   = 'INVALID ERTS SCENE IDENTIFICATION')
   IF(NBERTS(I).EQ.MAXINT) GO TO 320
```

310 CONTINUE

C

C

C IF LANDSAT-3 SET DATA SEQ. VERSION AND NUMBER OF CHANNELS

C

```
320 IF(NBERTS(1).EQ.3) LU3SEQ(2) = '1'
   IF(NBERTS(1).EQ.3) NERCHA = 5
```

C

```
IF(INERCOR.NE.'S') CALL MDFATL(
= ' '.CBS4CS(INERCOR.1.1).CORRECTED TAPE: UNSUPPORTED')
IF( INERCOR.EQ.'S'.AND.(NERGEO.NE.'BAD'.AND.NERGEO.NE.'ERT') )
= CALL MDNOTE(
= 'SYSTEM CORRECTED TAPE CANNOT BE'.CBS4CS(NERGEO.1.3).
= ' PROJECTION -- ERTS GEOMETRY ASSUMED')
IF( INERCOR.EQ.'3'.AND.(NERGEO.NE.'BAD') ) NERGEO = 'ERT'
```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

GP301P  
009

```

C
C
C DETERMINE MODE AND GAIN VALUES FOR BANDS (BYTE 30)
C
    MODTMP = MODTMP / 4
    MOECON = 0
    IF(MOD(MODTMP,2).NE.0) MOECON = 1
C
    DO 321 I = 1,NERCHA
        CAL CST4IN(NERBAN(I),1,NC4NI(1), 1+3,1)
        NERGA1(I) = 'L'
321 CONTINUE
C
    MODTMP = MODTMP / 2
    IF(MOD(MODTMP,2).NE.0) NERGA1(2) = 'H'
    MODTMP = MODTMP / 2
    IF(MOD(MODTMP,2).NE.0) NERGA1(1) = 'H'
C
    MODTMP = MODTMP/2
    DO 322 I = 1,NERCHA
        NERTHO(I) = '1'
        IF(1.0T.3) GO TO 322
        IF(MOD(MODTMP,2).NE.0) NERTHO(I) = '2'
322 CONTINUE
C
    IF(NERTHO(1).EQ.'1'.AND.MOZCOM.EQ.1) CALL MOFATL(
    - ' DECOMPRESSION OF LINEAR TRANSMISSION DATA NOT VALID')
C
    IF(NERTHO(1).EQ.'2'.AND.MOECON.EQ.0) CALL MOFATL(
    - ' UNSUPPORTED: COMPRESSION TRANS. MODE NOT DECOMPRESSED')
C
C
C
C INITIALIZE READ PARAMETERS FOR GODDARD FORMAT
C
    LU3LBF = 0      8 LINE 0(HEADER) IN BUFFER
    LU3RBF = 0      8 RECORD 0(HEADER) IN BUFFER
    LU3WIB = SZ3BUF  8 NUMBER OF WORDS PER PHYSICAL TAPE BLOCK
    LU3BIL = 1      8 NUMBER OF PHYSICAL TAPE BLOCKS PER SCAN LINE
    LU3BIB = NB4NI(SZ3BUF) 8 # OF BYTES PER PHYSICAL TAPE BLOCK
C
    NERLIN = 2340
    CTRLIN = (NERLIN + 1)*0.5
    CTRSAM = (NERSAM + 1)*0.5
C
    NSAIWW(WLIN,WMIN) = 1
    NSAIWW(WLIN,WMAX) = 2340
    NSAIWW(WSAM,WMIN) = NERSAM*(NCCT - 1)/NCCTOT + 1
    NSAIWW(WSAM,WMAX) = NERSAM*NCCT/NCCTOT
C
C
C PRINT REC IF TRACE IS ON OR DIAGNOSTIC COUNT IS CHANGED
C
    IF( (NTRACE.EQ.0).AND.(NDSAVE.EQ.NDTOTL) ) GO TO 900
    DO 800 LOWCHR = 1,144,48
        LOWINT = NI4NC(LOWCHR)

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**OP381P  
010**

```
      CALL ERPRNT(1,N14NC(40), KTABLE(LOWINT))  
000 CONTINUE  
C  
C  
000 RETURN  
C  
C  
      END
```

**OP306K**  
**001**

```

C
C
C HISTORY
C -----
C
C MARY TOMPKINS      LEC      08/11/79      REQUIREMENTS
C MARY TOMPKINS      LEC      08/25/79      STUBBED
C MARY TOMPKINS      LEMSCO  06/16/80      ADAPTED FROM OPN12N
C
C METHOD
C -----
C
C READ HEADER SECTORS FROM DISK FILE INTO LOCAL BUFFER.
C LOAD KOMNER. KOMKLS. KOMFIT. KOMDET FROM LOCAL BUFFER.
C DEFINE INPUT WINDOW. NUMBER OF CHANNELS. AND LENGTH OF RECORD.
C
C MACHINE-DEPENDENT CODE
C -----
C
C NONE.
C
C EXTERNAL REFERENCES
C -----
C
C MDFATL      & PRINT/COUNT/LOG 'FATAL ERROR' DIAGNOSTIC MESSAGE
C ERION       & INITIATE I/O & WAIT FOR COMPLETION
C
C EXCEPTIONS
C -----
C
C 1. THE FOLLOWING CONDITION GENERATE THE DIAONOSTICS SHOWN:
C
C          CONDITION                                DIAGNOSTICS
C          BAD STRIP NUMBER                          MDFATL
C
C GLOBAL DECLARATIONS
C -----
C
C INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES. COUNTERS
C INCLUDE KOHLU3.LIST      & COMMON I/O PACKET/POINTERS FOR UNIT 3
C INCLUDE KOMIO.LIST       & COMMON I/O FUNCTIONS
C INCLUDE KOMNER.LIST      & COMMON ERTS SCENE PARAMETERS
C INCLUDE KOMKLS.LIST      & COMMON CLASSIFICATION INFO
C INCLUDE KOMFIT.LIST      & COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C INCLUDE KOMDET.LIST      & TEMPORARY COMMON DISK FILE WINDOW. RECORD LENGTH
C INCLUDE WINDEF.LIST      & DEFINE STRUCTURE OF WINDOW PACKETS
C INCLUDE KOMIWW.LIST      & COMMON INPUT WINDOW PACKETS
C INCLUDE TRFORM.LIST      & DEFINE COORDINATE TRANSFORMATION FUNCTIONS

```

BAM PACKAGE APPENDIX N  
UTILITY ROUTINES

OP3DSK  
002

```

C
C LOCAL DECLARATIONS
C -----
C
C     INTEGER KBUFR(20,9)      8 INTERNAL WORK BUFFER FOR DISK FILE HEADER
C
C
C PROCEDURE
C -----
C
C     CALL TRACE('OPN12N')
C
C
C READ HEADER (SECTORS 0 THRU 6) INTO BUFFER
C
C     CALL RDHDR(KBUFR)
C     IF((KBUFR(4,1)).LT.1).OR.(KBUFR(4,1).GT.4)) CALL M0FATL(
C       - 'BAD STRIP NUMBER IN FILE 3 HEADER')
C     IF(M0FATL.NE.0) GO TO 900
C
C
C INITIALIZE DATA SEQUENCE/VERSION & RECORD FORMAT/VERSION
C
C     LU3SEQ(1)='DSK'
C     LU3SEQ(2)='0'
C     LU3REF(1)='PXB'
C     LU3REF(2)='0'
C
C
C
C LOAD SECTORS 0 THRU 6 INTO KOMNER, KOMKLS, AND KOMFIT
C
C     DO 411 NWD=1,SIZNER
C       411 KOMNER(NWD)=KBUFR(NWD,1)
C     DO 412 NWD=1,SIZKLS
C       412 KOMKLS(NWD)=KBUFR(NWD,4)
C     DO 413 NWD=1,SIZFIT
C       413 KOMFIT(NWD)=KBUFR(NWD,6)
C
C
C
C LOAD SECTORS 7 & 8 INTO KOMDET
C
C     DO 460 NWD=1,SIZDET
C       KOMDET(NWD)=KBUFR(NWD,8)
C     460 CONTINUE
C
C
C
C DEFINE INPUT WINDOW FROM KOMDET WINDOW
C
C     HSA1WW(WLIN,WMIN)=MSADWW(WLIN,WMIN,NCCT)
C     HSA1WW(WLIN,WMAX)=MSADWW(WLIN,WMAX,NCCT)
C     HSA1WW(WSAM,WMIN)=MSADWW(WSAM,WMIN,NCCT)
C     HSA1WW(WSAM,WMAX)=MSADWW(WSAM,WMAX,NCCT)
C
C
C     IF(MERGE0.NE.'ERT') GO TO 470
C
C     - COR1WW(WLIN,WMIN)=CORL4A(HSA1WW(WLIN,WMIN),HSA1WW(WSAM,WMIN))

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

OP3DSK  
003

```

CORINH(WLIN,WMAX)=CORL4A(MSAINH(WLIN,WMAX),MSAINH(MSAM,WMAX))
CORINH(MSAM,WMIN)=CORS4A(MSAINH(WLIN,WMIN),MSAINH(MSAM,WMIN))
CORINH(MSAM,WMAX)=CORS4A(MSAINH(WLIN,WMAX),MSAINH(MSAM,WMAX))
GO TO 490
470 CORINH(WLIN,WMIN)=MSAINH(WLIN,WMIN)
CORINH(WLIN,WMAX)=MSAINH(WLIN,WMAX)
CORINH(MSAM,WMIN)=MSAINH(MSAM,WMIN)
CORINH(MSAM,WMAX)=MSAINH(MSAM,WMAX)
490 CONTINUE
C
C
C DEFINE LENGTH OF DISK RECORD IN UNIVAC SECTORS FROM KOMDET
C
LU3LRS=LDETR5(NCCT)
C
C
C DEFINE NUMBER OF INPUT DISK CHANNELS AVAILABLE
C
IF(NDIFATL.EQ.0) MERCHA=NOUTCH
C
C
C ZERO KOMDET
C
DO 800 NWD=1,SIZEDET
KOMDET(NWD)=0
800 CONTINUE
C
C
C DONE
C
900 RETURN
C
C
C
C
C
C
INTERNAL
SUBROUTINE RMDR(KBUFR)
IOSIZE(LU3PKT)=252      3 HEADER SIZE IN WORDS FOR OP3DSK
IOADDR(LU3PKT)=LOC(KBUFR)
IOSECT(LU3PKT)=0
IOFUNC(LU3PKT)='8K'      3 READ HEADER
CALL ERIOH(LU3PKT)
RETURN
END

```



DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

OP3MOP  
001

SUBROUTINE OP3MOP 3 OPEN INPUT MOP TAPE

```

C
C
C
C HISTORY
C -----
C
C   MARY TOMPKINS   LEC   08/08/79   REQUIREMENTS
C   MARY TOMPKINS   LEC   10/15/79   ALGORITHM DESIGN
C   MARY TOMPKINS   LEC   10/31/79   ALGORITHM CODING
C   MARY TOMPKINS   LEMSCO 09/28/79   NTEMP=0 WHEN CHANGING NOXFPI
C
C
C METHOD
C -----
C
C   CALL O3TDR. CALL O3HOR. IF NOANCL > 0 CALL O3ANCL AND
C   PROCESS NOANCL ANCILLARY RECORDS. THEN CALL O3ANOT.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C   NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C   O3TDR   3 PROCESS TAPE DIRECTORY RECORD
C   O3HOR   3 PROCESS HEADER RECORD FROM MOP TAPES
C   O3ANCL  3 PROCESS ANCILLARY RECORDS FROM MOP TAPES
C   O3ANOT  3 PROCESS ANNOTATION RECORDS FROM MOP TAPES
C   CBS4CS  3 VARIABLE CHARACTER STRING FOR FIXED CHAR STRING
C             DOUBLE PRECISION CBS4CS
C
C
C GLOBAL DECLARATIONS
C -----
C
C   INCLUDE KONXQT.LIST   3 COMMON PROGRAM EXECUTION SWITCHES. COUNTERS
C   INCLUDE KOMTBL.LIST   3 COMMON BLOCKS AND DEFINE PROCEDURES
C   INCLUDE KOMLU3.LIST   3 PACKET/POINTERS FOR UNIT 3
C   INCLUDE FIDF.LIST     3 DEFINES STRUCTURE OF LU3FID ARRAY
C
C
C LOCAL DECLARATIONS
C -----
C
C   INTEGER NOXREF   3 INDEX FOR LU3REF IN NIVOLM/NIVOLR
C   INTEGER NOXFPI   3 INDEX FOR FPI OF TAPE IN NIVOLM/NIVOLR
C   INTEGER NOXCHA   3 INDEX FOR NUMBER OF CHANNELS IN NIVOLM
C   INTEGER NIVOLM(5,2,2,2) / 3 * RECORDS/VOLUME ON MSS 811 TAPE
C                               (1.   (2.   (3.   (4.   (5.
C (VOL.AM. 000FPI,4CHAN)      .1,1,1)
C                               4800. 4800. 0000. 0000. 0000.

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

OP3NDP  
002

```

C (VOL.PH. 800FPI.4CHAN)                .2.1.1)
      &                3976.  3976.  3980.  0000.  0000.
C (VOL.AM.1600FPI.4CHAN)                .1.2.1)
      &                9600.  0000.  0000.  0000.  0000.
C (VOL.PH.1600FPI.4CHAN)                .2.2.1)
      &                5964.  5966.  0000.  0000.  0000.
C (VOL.AM. 800FPI.5CHAN)                .1.1.2)
      &                4000.  4000.  4000.  0000.  0000.
C (VOL.PH. 800FPI.5CHAN)                .2.1.2)
      &                4970.  4970.  4975.  0000.  0000.
C (VOL.AM.1600FPI.5CHAN)                .1.2.2)
      &                6000.  6000.  0000.  0000.  0000.
C (VOL.PH.1600FPI.5CHAN)                .2.2.2)
      &                7455.  7460.  0000.  0000.  0000 /
C
C
      INTEGER NIVOLR(2,2,2) /      & * RECORDS/VOLUME ON RBV TAPE
      (1.      (2.
C (VOL.AR. 800FPI)                .1.1)
      &                2062.  2063.
C (VOL.PR. 800FPI)                .2.1)
      &                2661.  2661.
C (VOL.AR.1600FPI)                .1.2)
      &                4125.  0000.
C (VOL.PR.1600FPI)                .2.2)
      &                5322.  0000 /
C
C
C PROCEDURE
C -----
C
      CALL TRACE
C
      CALL O3TOR
      CALL O3HDR(NOANCL)
      IF (NOANCL.GT.0) CALL O3ANCL(NOANCL)
      CALL O3ANOT
C
C INITIALIZE REEL IDENTIFIER FOR FIRST 2 VOLUMES
C
      IF (LU3FID(FIDCRL).NE.' ') LU3REL(1)=LU3FID(FIDCRL) &CURRENT REEL
      IF (LU3FID(FIDNRL).NE.' ') LU3REL(2)=LU3FID(FIDNRL) &NEXT REEL
C
C
C INITIALIZE LOW-HIGH RECORD * FOR EACH TAPE VOLUME
C
      IF (LU3REF(1).EQ.'AM' .OR. LU3REF(1).EQ.'PH') GO TO 150
      IF (LU3REF(1).EQ.'AR' .OR. LU3REF(1).EQ.'PR') GO TO 300
      CALL MDATL ( 'OP3NDP DETECTS INVALID RECORD FORMAT = ',
      & C094C8(LU3REF(1),(1),3) )
      GO TO 900
C
C
C
C NSS TAPE
C
      150 IF (LU3SEQ(1).EQ.'B50') GO TO 230

```

**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**OPINOP  
003**

```

C
C
C INDEX FOR NIVOLM ARRAY - DATA SEQUENCE
C
      NOXREF=1                      0 'AM'
      IF (LU3REF(1).EQ.'PH') NOXREF=2  0 'PH'
C
C
C ASSUME THAT THIS TAPE, IF COPIED, HAS SAME DENSITY AS ORIGINAL
C
      NOXFPI=1                      0 800 FPI
      IF (LU3FID(FIOFPI).EQ.'1600') NOXFPI=2  0 1600 FPI
C
C
C NUMBER OF CHANNELS
C
      NOXCHA=1                      0 4 CHANNELS
      IF (NERCHA.EQ.5) NOXCHA=2      0 5 CHANNELS
C
C
C DETERMINE (DENSITY DEPENDENT) LOWEST & HIGHEST RECORD ON EACH VOLUME
C
      100 CONTINUE
          NTEMP=0
          DO 200 I=1,V3HAX
              IF (NIVOLM(I,NOXREF,NOXFPI,NOXCHA).EQ.0) GO TO 210
              LU3RLO(I)=NTEMP+1
              NTEMP=NIVOLM(I,NOXREF,NOXFPI,NOXCHA) + NTEMP
              LU3RHI(I)=NTEMP
      200 CONTINUE
      210 CONTINUE
C
C      IF HIGHEST VOLUME FOR THIS DENSITY HAS NO RECORDS THEN
C      THIS TAPE MUST BE A 1600 FPI COPY (HOPEFULLY REEL FOR REEL)
C      OF AN 800 FPI ORIGINAL
C
      IF (LU3RLO(LU3VHI).NE.0) GO TO 220
      NOXFPI=1      0 CHANGE DATA FORMAT TO 800 FPI FROM 1600 FPI
      CALL MODARN(
-      'TAPE IS 1600 FPI COPY OF 800 FPI ORIGINAL')
      GO TO 100
C
C
      220 IF (LU3RLO(LU3VHI+1).NE.0) CALL MODATL(
-      'TAPE IS 800 FPI COPY OF 1600 FPI ORIGINAL')
      GO TO 900
C
C
C DUE TO DIFFERENCE IN STRUCTURE OF 'BIL' AND 'BSQ' RECORDS. AT
C THIS TIME SUPPORT FOR ONLY CHANNEL 1 'BSQ' IS PROVIDED.
C CHANNEL 1 IS ALWAYS ON ONE VOLUME.
C
      230 LU3RLO(1)=1
          IF (LU3REF(1).EQ.'AM') LU3RHI(1)=2400
          IF (LU3REF(1).EQ.'PH') LU3RHI(1)=2903
          GO TO 900

```

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

OP3NDP  
884

```

C
C
C RBY TAPE
C
  300 NOXREF=1                & 'AR'
    IF (LU3REF(1).EQ.'PR') NOXREF=2 & 'PR'
    NOXFPI=1                & 800 FPI
    IF (LU3FID(FIDFPI).EQ.'1800') NOXFPI=2 & 1800 FPI
C
C DETERMINE (DENSITY DEPENDENT) LOWEST & HIGHEST RECORD ON EACH VOLUME
C
  310 CONTINUE
    NTEMP=0
    DO 320 I=1,V3MAX
      IF (NIVOLR(1,NOXREF,NOXFPI).EQ.0) GO TO 330
      LU3RLO(I)=NTEMP+1
      NTEMP=NIVOLR(1,NOXREF,NOXFPI)+NTEMP
      LU3RHI(I)=NTEMP
    320 CONTINUE
  330 CONTINUE
C
C IF HIGHEST VOLUME FOR THIS DENSITY HAS NO RECORDS THEN
C THIS TAPE MUST BE A 1800 FPI COPY (HOPEFULLY REEL FOR REEL)
C OF AN 800 FPI ORIGINAL
C
    IF (LU3RLO(LU3VHI).NE.0) GO TO 350
    NOXFPI=1 & CHANGE DATA FORMAT TO 800 FPI FROM 1800 FPI
    CALL MDHARN(
      'TAPE IS 1800 FPI COPY OF 800 FPI ORIGINAL')
    GO TO 310
C
C
C 350 IF (LU3RLO(LU3VHI+1).NE.0) CALL MDFATL(
      'TAPE IS 800 FPI COPY OF 1800 FPI ORIGINAL')
C
C 800 RETURN
C
  END

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

OP3TAP  
001

SUBROUTINE OP3TAP 3 OPEN INPUT ERTS NSS TAPE ON UNIT 3

```

C
C
C
C HISTORY
C -----
C
C   MARY TOMPKINS      LEC   08/02/79   REQUIREMENTS
C   J C CRISP          LEC   09/20/79   ALGORITHM DESIGN
C   J C CRISP          LEC   09/20/79   ALGORITHM CODING
C
C
C METHOD
C -----
C
C   SET KTBLY = ' '. READ FIRST PART OF NEXT TAPE RECORD INTO KTABLE.
C   REPOSITION TAPE TO BEGINNING OF THIS RECORD.
C   IF RECORD LENGTH > ALLOWABLE RANGE OF BIP FORMAT ID RECORD, CALL OPMDP
C   ELSE CALL OP3BIP.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C   NONE
C
C
C EXTERNAL REFERENCES
C -----
C
C   ERION      3 INITIATE I/O AND WAIT FOR COMPLETION
C   OP3MDP     3 OPEN INPUT MDP TAPE
C   OP3BIP     3 OPEN INPUT ERTS NSS TAPE
C   ERTWAT     3 TIMED WAIT
C   MDPATL     3 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C   MDNOTE     3 PRINT/LOG/ 'PRINT' MESSAGES
C
C
C EXCEPTIONS
C -----
C
C   1. BEFORE CALLING THIS SUBROUTINE THE TAPE MUST BE POSITIONED AT THE
C       ID RECORD FOR TAPES IN BIP FORMAT AND AT THE TAPE DIRECTORY RECORD
C       FOR TAPES FROM MASTER DATA PROCESSOR.
C
C   2. THE FOLLOWING CONDITIONS GENERATE THE DIONOSTICS SHOWN:
C
C       'EOF' I/O STATUS      FATAL ERROR
C       'BADP' I/O STATUS     FATAL ERROR
C       'LOST' I/O STATUS     NOTE
C       'BADR' I/O STATUS     NOTE
C
C
C GLOBAL DECLARATIONS
C -----
C

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**OP3TAP  
002**

```

      INCLUDE KONXGT.LIST      & COMMON PROGRAM EXECUTION SWITCHES,COUNTERS
      INCLUDE KONLU3.LIST      & PACKET/POINTERS FOR UNIT 3
      INCLUDE KONTEL.LIST      & COMMON BLOCKS AND DEFINE PROCEDURES
      INCLUDE KONIO .LIST      & FORTRAN MANIPULATION OF ASH I/O PACKETS
      INCLUDE FIDEF.LIST      & DEFINES STRUCTURE OF LU3FID ARRAY

C
C
C LOCAL DECLARATIONS
C -----
C
      INTEGER NMDSRD      & NUMBER OF WORDS READ FROM RECORD
C
C
C PROCEDURE
C -----
C
      CALL TRACE
C
C
C CHECK TAPE DATA TRANSFER FORMAT
C
      IF(LU3FID(FIDBFH).EQ.'BB'.OR.LU3FID(FIDBFH).EQ.'BST') GO TO 200
      CALL MDATL( 'BAD TAPE DATA TRANSFER FORMAT ON JASO')
      GO TO 900
C
C
C FLAG PREVIOUS CONTENTS OF KTABLE AS DESTROYED
C
      200 KTBLY = . . .
C
C
C READ 20 WORDS OF RECORD INTO KTABLE
C
      IADDR(LU3PKT) = LOC(KTABLE)
      ISIZE(LU3PKT) = 20
      IMAIT(LU3PKT) = 0
      IOFUNC(LU3PKT) = 'BK'      & READ FORWARD
      CALL ERION(LU3PKT)
      NMDSRD = IONMDS(LU3PKT)
C
C
C CHECK STATUS OF READ
C
      ISTAT = ICODE(LU3PKT)
      IF( (ISTAT.EQ.'EOF').OR.(ISTAT.EQ.'BADF') )CALL MDATL(
      = CDS4CS(ISTAT,1,4), ' WHILE READING FIRST RECORD')
C
      IF( (ISTAT.EQ.'LOST').OR.(ISTAT.EQ.'BADR') )CALL MDNOTE(
      = CDS4CS(ISTAT,1,4), ' WHILE READING FIRST RECORD')
C
C
C REPOSITION TAPE TO BEGINNING OF RECORD
C
      IOFUNC(LU3PKT) = 'BL'      & READ BACKWARD
      CALL ERTHAT(2000)          & REVERSE TAPES GENTLY
      CALL ERION(LU3PKT)

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

OP3TAP  
003

```
C
C
C IF RECORD IS AT LEAST 20 WORDS LONG. TAPE IS NDP
C      IF(NWORDS.GE.20) CALL OP3NDP
C
C
C ELSE. TAPE IS BIP
C      IF(NWORDS.LT.20) CALL OP3BIP
C
C
C 900 RETURN
C
C      END
```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

OZANCL  
001

```

SUBROUTINE OZANCL(      0 PROCESS ANCILLARY RECORDS FROM NDP
      1 NOANCL)      0 NUMBER OF ANCILLARY RECORDS
      -----
C
C
C HISTORY
C -----
C
C      MARY TOMPKINS      LEC      00/00/70      REQUIREMENTS
C      MARY TOMPKINS      LEC      10/20/70      ALGORITHM DESIGN
C      MARY TOMPKINS      LEC      11/04/70      ALGORITHM CODING
C
C
C METHOD
C -----
C
C      DEFINE ANCLSZ PARAMETER. IF KTBLSZ<ANCLSZ GENERATE FATAL ERROR.
C      RETURN. PROCESS NOANCL ANCILLARY RECORDS INDIVIDUALLY INTO KTABLE.
C      IF RECORD INDICATOR INDICATES RECORD IS AN ANNOTATION RECORD
C      REPOSITION TAPE TO BEGINNING OF RECORD AND RETURN. (THE ANCILLARY
C      RECORDS ARE CURRENTLY BYPASSED AND THE USER ACCOMPLISHES GEOMETRIC
C      CORRECTION THROUGH HIS OWN CONTROL NETWORK. IN THE FUTURE THESE
C      RECORDS MAY BE USED FOR AUTOMATIC GEOMETRIC CORRECTION.)
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      BST400      0 INTERNAL BYTE STRING FOR 0-BIT EXTERNAL BYTE STRING
C      ERTWAT      0 TIMED WAIT
C      ERION      0 INITIATE I/O AND WAIT FOR COMPLETION
C      GETBYT      0 GET NON-NEGATIVE INTEGER FROM BYTE STRING
C      ERTWAT      0 TIMED WAIT
C      NDPATL      0 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C      INTEGER N04N1      0 # OF BYTES FOR # OF INTEGERS
C
C
C EXCEPTIONS
C -----
C
C      1. IF KTBLSZ < ANCLSZ ISSUE FATAL ERROR
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KONXGT.LIST      0 COMMON PROGRAM EXECUTION SWITCHES,COUNTERS
C      INCLUDE KONTBL.LIST      0 COMMON BLOCKS AND DEFINE PROCEDURES
C      INCLUDE KOMLUS.LIST      0 PACKET/POINTERS FOR UNIT 3

```



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

03ANCL  
002

```

      INCLUDE KONIO.LIST      & FORTRAN MANIPULATION OF ASH I/O PKTS
C
C
C LOCAL DECLARATIONS
C -----
C
      INTEGER 0044/0044/      & MNEMONICS FOR OCTAL 0044
      INTEGER NUMRDS          & NUMBER OF READS
      PARAMETER ANCLSZ = 3

C
C
C PROCEDURE
C -----
C
      CALL TRACE

C
C
C CHECK BUFFER SIZE
C
      IF(KTBLSZ.LT.ANCLSZ) CALL HDFATL( 'KTBLSZ < ANCLSZ')
      IF(KTBLSZ.LT.ANCLSZ) GO TO 900
      NUMRDS = 1

C
C
C PROCESS NUMBER OF AVAILABLE ANCILLARY RECORDS
C
      IOADDR(LU3PKT) = LOC(KTABLE)
      IOSIZE(LU3PKT) = ANCLSZ
      IOWAIT(LU3PKT) = 10

C
      DO 110 NANCL = 1,NOANCL
        IOFUNC(LU3PKT) = 'BK' & READ FORWARD
        CALL ERLOW(LU3PKT)
        ISTAT = ICODE(LU3PKT)
        IF(ISTAT.EQ.'EOF')NUMRDS = 3 & READ PAST ANNOT. REC
        IF(ISTAT.EQ.'EOF') GO TO 120
      110 CONTINUE

C
C
C IF BUFFER CONTENTS ARE IN BB FORMAT. THEN CONVERT THEM TO BST
C
      120 IF (LU3BFH.EQ.'BB') CALL BST4BB (KTABLE, KTABLE.NB4NI(ANCLSZ))

C
C
C CHECK IF READ CONTINUED INTO ANNOTATION RECORD
C
      DO 500 I = 1,NUMRDS
        CALL GETBYT(NUMREC, KTABLE.6) & RECORD TYPE NUMBER
        IF (NUMREC.EQ.0044) GO TO 900
        IOFUNC(LU3PKT) = 'BL' & READ BACKWARD
        CALL ERTHAT(2000) & REVERSE TAPE GENTLY
        CALL ERLOW(LU3PKT)
      500 CONTINUE

C
C
      900 RETURN

```

**OAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**03ANCL  
003**

**END**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

03ANOT  
001

SUBROUTINE 03ANOT      0 READ ANNOTATION RECORD FROM TAPES

```

C
C
C
C HISTORY
C -----
C
C      J C CRISP          LEC      08/08/79      REQUIREMENTS
C      MARY TOMPKINS     LEC      10/27/79      ALGORITHM DESIGN
C      MARY TOMPKINS     LEC      11/09/79      ALGORITHM CODING
C      MARY TOMPKINS     LENSCH 01/18/80      SEPARATE 03SZAH/..PH/..AR/..PR
C
C
C METHOD
C -----
C
C      DEFINE VIDEOSZ AND ANOTSZ PARAMETERS.  IF KTBSZ < ANOTSZ GENERATE FATAL
C      ERROR AND RETURN.  READ ANOTSZ WORDS OF ANNOTATION RECORD INTO KTABLE.
C      CHECK RECORD TYPE TO VERIFY ANNOTATION RECORD - IF ANCILLARY RECORD.  READ
C      NEXT RECORD ELSE.  IF NOT ANNOTATION RECORD.  ISSUE FATAL MESSAGE & RETURN.
C      IF LU3BFH IS '88' CALL BST488.  CALL CST4AS TO CONVERT ANNOTATION RECORD
C      TO CHARACTER STRING.  EXTRACT/DECODE CHARACTER STRING INFORMATION.
C      VERIFY AND COMPARE INFORMATION FROM HEADER AND ANNOTATION RECORDS.
C      CALL APPROPRIATE 03SZ-- ROUTINE TO SET NERLIN. NERSAM AND INITIALIZE
C      INPUT WINDOW PACKETS.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      MOFATL      0 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C      ERPRNT      0 PRINT IMAGE ON TTY OR LINE PRINTER (CHARACTER)
C      GETBYT      0 GET ONE NON-NEG INTEGER FROM BYTE STRING
C      CST4AS      0 CHARACTER STRING FOR ASCII BYTE STRING
C      GETDBY      0 GET NON-NEG INTEGER FROM DOUBLE BYTE STRING FROM BYTE STRING
C      ERTWAT      0 TIMED WAIT
C      GETNYB      0 GET NON-NEG INTEGER FROM NYBLE IN BYTE STRING
C      MOHARN      0 PRINT/LOG/COUNT 'WARNING' MESSAGES
C      MOVCHR      0 MOVE A CHARACTER FROM ONE STRING TO ANOTHER
C      DCODE      0 DECODE NUMERIC CHARACTER STRING
C      GETCHR      0 GET CHARACTER FROM CHARACTER STRING
C      MOVCS      0 MOVE SUBSTRING 2 TO SUBSTRING 1
C      ERIOW      0 INITIATE I/O AND WAIT FOR COMPLETION
C      BST488      0 INTERNAL BYTE STRING FOR 8-BIT EXTERNAL BYTE STRING
C      MDNOTE      0 PRINT/LOG 'NOTE' MESSAGES
C      ERIOW      0 INITIATE I/O AND WAIT FOR COMPLETION
C      03SZAH      0 OBTAIN SCENE SIZE & INPUT WINDOW ('AH' TAPES)
C      03SZPH      0 OBTAIN SCENE SIZE & INPUT WINDOW ('PH' TAPES)
C      03SZAR      0 OBTAIN SCENE SIZE & INPUT WINDOW ('AR' TAPES)
C      03SZPR      0 OBTAIN SCENE SIZE & INPUT WINDOW ('PR' TAPES)

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

03ANOT  
002

C     CST4IN        & CHARACTER STRING FOR INTEGER  
       DOUBLE PRECISION CDS4CS   & VARIABLE LENGTH CHAR STRING FOR FIXED LENGTH CH  
       INTEGER NI4NC            & # OF INTEGERS FOR # CHARACTERS  
       INTEGER NC4NI            & # OF CHARS FOR # OF INTEGERS  
       INTEGER LCHREQ           & LOCATION OF CHAR IN STRING - SEARCH CHAR  
       INTEGER NB4NI            & # OF BYTES FOR # OF INTEGERS  
       REAL        DEO           & DEGREES FOR DEG.MIN.SEC

C  
C  
C EXCEPTIONS  
C -----

C     1. THE FOLLOWING CONDITIONS GENERATE THE DIAGNOSTICS SHOWN:  
       CONFLICT IN SCENE NUMBER       WARNING  
       DATA ERROR IN ANOT REC        FATAL ERROR  
       INVALID ERTS NUMBER            FATAL ERROR  
       TAPE NOT AT ANCILLARY OR  
       ANNOTATION RECORD               FATAL ERROR  
       KTBLSZ < ANOTSZ                FATAL ERROR  
       'EOF' I/O STATUS                FATAL ERROR  
       'BADF' I/O STATUS               FATAL ERROR  
       'LOST' I/O STATUS               NOTE  
       'BADR' I/O STATUS               NOTE

C  
C  
C GLOBAL DECLARATIONS  
C -----

C     INCLUDE KOMXQT.LIST            & COMMON PROGRAM EXECUTION SWITCHES, COUNTERS  
       INCLUDE KOHLU3.LIST           & PACKET/POINTERS FOR UNIT3  
       INCLUDE KOMNER.LIST           & ERTS SCENE PARAMETERS  
       INCLUDE KOMTBL.LIST           & COMMON MULTI-PURPOSE TABLE  
       INCLUDE KOMIO .LIST           & FORTRAN ALLOWING USE OF ASSEMBLER I/O PACKETS  
       INCLUDE MAXINT.LIST           & MAX INTEGER CONTAINED IN MACHINE  
       INCLUDE WINDEF.LIST           & DEFINE STRUCTURE OF WINDOW PACKET  
       INCLUDE KOMIWH.LIST           & COMMON INPUT WINDOW PACKETS

C  
C  
C LOCAL DECLARATIONS  
C -----

      INTEGER NOSAVE            & TEMP TO STORE # OF PREVIOUS DIAGNOSTIC  
       LOGICAL BAD               & TRUE IF CONTENTS OF SOME VARIABLE ARE INVALID  
       INTEGER IRECNT            & COUNT OF RECORDS READ  
       INTEGER ISTAT             & I/O READ STATUS  
       INTEGER O333/O333/        & MNEMONIC FOR OCTAL333 (ANOT REC TYP)  
       INTEGER NTERYS(3)         & TEMP. FOR ERTS SCENE IDENTIFICATION  
       INTEGER NRECTY            & RECORD TYPE  
       INTEGER LATD               & SCENE CENTER LATITUDE DEGREES  
       INTEGER LATM               & SCENE CENTER LATITUDE MINUTES  
       INTEGER LOND               & SCENE CENTER LONGITUDE DEGREES  
       INTEGER LONM               & SCENE CENTER LONGITUDE MINUTES  
       INTEGER NLATD              & NADIR LATITUDE DEGREES  
       INTEGER NLATM              & NADIR LATITUDE MINUTES  
       INTEGER NLOND              & NADIR LONGITUDE DEGREES  
       INTEGER NLONM              & NADIR LONGITUDE MINUTES

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

03ANOT  
003

```

INTEGER MODE          & DATA MODE '1' = LINEAR, '2' = COMPRESSED
INTEGER IOAIN          & GAIN OPTION: 'H' - HIGH, 'L' - LOW
INTEGER LOCEO          & LOCATION IN IMAGE OF MERGED
INTEGER IMAGEO(6) // 'BAD', 'LCC', 'PS', 'SON', 'UTH', 'HOM' /
INTEGER IMGO // 'LPSUM' /      & GODDARD IMAGE GEOMETRY
PARAMETER ANOTSZ = 100

```

C  
C  
C  
C  
C  
C

PROCEDURE

-----

C  
C  
C

CALL TRACE

C

```

LU3CBR = 0
LU3HBR = 2      & ALLOWABLE # OF 'BADR' BEFORE 'BADR' = 'BADF'
NDSAVE = NDTOTL

```

C  
C  
C  
C

CHECK BUFFER SIZE

```

IF (KTBSZ.LT.ANOTSZ) CALL MDFATL( 'KTBSZ < ANOTSZ')
IF (KTBSZ.LT.ANOTSZ) GO TO 900

```

C  
C  
C  
C  
C  
C

READ UP TO 3 RECORDS UNTIL KTABLE BUFFER  
CONTAINS ANNOTATION RECORD

```

DO 120 IRECNT = 1,3
  IOADDR(LU3PKT) = LOC(KTABLE)
  IOSIZE(LU3PKT) = ANOTSZ
  IOWAIT(LU3PKT) = 10
  IOFUNC(LU3PKT) = 'BK' & READ FORWARD
  CALL ERLOW(LU3PKT)
  ISTAT = ICODE(LU3PKT)
  IF ( (ISTAT.EQ.'EOF').OR.(ISTAT.EQ.'BADF') ) CALL MDFATL(
- CBS4CS(ISTAT,1,4), ' WHILE READING ANNOTATION RECORD')
  IF ( (ISTAT.EQ.'BADR').OR.(ISTAT.EQ.'LOST') ) CALL MDNOTE(
- CBS4CS(ISTAT,1,4), ' WHILE READING ANNOTATION RECORD')
  IF (LU3BFH.EQ.'BB') CALL BST4BB(KTABLE,
- KTABLE,NB4NI(ANOTSZ)) & IF BB, TRANSFORM TO BST
  CALL GETBYT(NRECTY, KTABLE,6) & OBTAIN RECORD TYPE
  IF (NRECTY.EQ.0333) GO TO 150 & ANNOTATION REC FOUND
  CALL MDNOTE( 'RECORD NOT AN ANNOTATION RECORD -- IGNORED')
120 CONTINUE
  CALL MDFATL( 'ANNOTATION RECORD NOT FOUND')
  GO TO 900

```

C  
C  
C  
C  
C  
C  
C  
C

CONVERT ANNOTATION RECORD TO CHARACTER STRING FROM ASCII

```

150 CALL CST4AS(KTABLE, KTABLE,NB4NI(ANOTSZ))

```

EXTRACT CHARACTER INFORMATION FROM ANNOTATION RECORD

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**03ANOT  
004**

```

CALL DCODE(NERDAY,REAL,KODTYP, KTABLE.7.2) 8 DAY EXPOSURE
IF(KODTYP.NE.'IN') NERDAY = 99
IF(NERDAY.LT.1.OR.NERDAY.GT.31)CALL MDWARN(
- 'BAD EXPOSURE DAY FROM ANNOTATION RECORD')
C
CALL MOVCST(NERMON.1.3, KTABLE.9.3.' ') 8 MONTH
C
CALL DCODE(NERYR,REAL,KODTYP, KTABLE.12.2) 8 YEAR
IF(KODTYP.NE.'IN')CALL MDWARN(
- 'BAD EXPOSURE YEAR FROM ANNOTATION RECORD')
C
CALL DCODE(LATD,REAL,KODTYP, KTABLE.18.2) 8 SCENE CENTER LAT-DEG
IF(KODTYP.NE.'IN')CALL MDATL(
- 'BAD SCENE CENTER DEG LAT FROM ANNOTATION RECORD')
C
CALL DCODE(LATH,REAL,KODTYP, KTABLE.21.2) 8 SCENE CENTER LAT-MIN
IF(KODTYP.NE.'IN')CALL MDWARN(
- 'BAD SCENE CENTER MIN LAT FROM ANNOTATION RECORD -- 30 ASSUMED')
IF(KODTYP.NE.'IN')LATH = 30
C
CALL DCODE(LOND,REAL,KODTYP, KTABLE.25.3) 8 SCENE CENTER LON-DEG
IF(KODTYP.NE.'IN')CALL MDATL(
- 'BAD SCENE CENTER DEG LON FROM ANNOTATION RECORD')
C
CALL DCODE(LONM,REAL,KODTYP, KTABLE.29.2) 8 SCENE CENTER LON-MIN
IF(KODTYP.NE.'IN')CALL MDWARN(
- 'BAD SCENE CENTER MIN LON FROM ANNOTATION RECORD -- 30 ASSUMED')
IF(KODTYP.NE.'IN')LONM = 30
C
CALL DCODE(NERSEL,REAL,KODTYP, KTABLE.74.2) 8 SUN ELEVATION
IF( (NERSEL.LT.0).OR.(NERSEL.GT.90) )KODTYP = 'ERR'
IF(KODTYP.NE.'IN')CALL MDWARN(
- 'BAD SUN ELEVATION FROM ANNOTATION RECORD')
C
CALL DCODE(NERSAZ,REAL,KODTYP, KTABLE.78.3) 8 SUN AZIMUTH
IF( (NERSAZ.LT.-360).OR.(NERSAZ.GT.360) )KODTYP = 'ERR'
IF(KODTYP.NE.'IN')CALL MDWARN(
- 'BAD SUN AZIMUTH FROM ANNOTATION RECORD')
C
C
HMYDEG = -9999
NLATD = -9999
NLATH = 0
NLOND = -9999
NLONM = 0
CALL MDNOTE( '1978 MDP FORMAT -- NO NADIR')
C
IF(LU3SEQ(1).EQ.'BIL') 00 TO 250
CALL GETCHR(MODE, KTABLE.92) 8DATA MODE
IF( (MODE.NE.'1').AND.(MODE.NE.'2') )CALL MDWARN(
- 'BAD DATA MODE FROM ANNOTATION RECORD')
C
CALL GETCHR(IGAIN, KTABLE.91) 8 GAIN OPTION
IF( (IGAIN.NE.'H').AND.(IGAIN.NE.'L') ) CALL MDWARN(
- 'BAD GAIN OPTION FROM ANNOTATION RECORD')
C

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**03ANOT  
005**

```

      DO 230 I = 1,NERCHA
        CALL CSTVIN(NERBAN(I),I,NCYN(I), I+3.1 )
        NEROA(I) = IOAIN
        NERTHO(I) = MODE
230    CONTINUE
C
250  CALL GETCHR(IGEO, KTABLE.84) & PROJECTION GEOMETRY
C
      LOCGEO = LCHREQ(IGEO,1.5,IGEO) + 1
      NERGEO = IMAEO(LOCGEO)
C
      CALL DCODE(NTERTS(1),REAL,KOOTYP, KTABLE.109.1) & ERTS PROJECT NO.
      IF(KOOTYP.NE.'IN')NTERTS(1) = MAXINT
C
      CALL DCODE(NTERTS(2),REAL,KOOTYP, KTABLE.110.4) & DAYS/LAUNCH
      IF(KOOTYP.NE.'IN')NTERTS(2) = MAXINT
C
      CALL DCODE(NTERTS(3),REAL,KOOTYP, KTABLE.115.5) & GMT
      IF(KOOTYP.NE.'IN')NTERTS(3) = MAXINT
C
      CALL GETCHR(NERCOR, KTABLE.82) & TYPE CORRECTION APPLIED
C
      CALL GETCHR(NERRES, KTABLE.86) & RESAMPLING ALGORITHM
C
C
C  CONVERT TO DEGREES FROM DEGREES & MINUTES
C
      CTRLAT = DEG(LATD,LATM,0.0)
      CTRLON = DEG(LOND,LONM,0.0)
      DIRLAT = DEG(NLATD,NLATM,0.0)
      DIRLON = DEG(NLOND,NLONM,0.0)
C
C
C  VERIFY BEST VALUE OF PROJECT NO.,DAY,TIME RELATIVE TO LAUNCH:
C
C  -- PROJECT NUMBER
C
300  IOUT = MAXINT
      BAD = ( (NTERTS(1).LT.1).OR.(NTERTS(1).GT.8) )
      IF(BAD) CALL MDWARN(
- 'BAD ERTS NUMBER FROM MDP ANNOTATION')
      IF(.NOT.BAD)IOUT = NTERTS(1)
      BAD = ( (NERTS(1).LT.1).OR.(NERTS(1).GT.8) )
      IF(BAD) CALL MDWARN(
- 'BAD ERTS NUMBER FROM MDP HEADER')
      IF(.NOT.BAD) IOUT = NERTS(1)
      NERTS(1) = IOUT
C
C
C  -- DAY RELATIVE TO LAUNCH AT OBSERVATION
C
      IOUT = MAXINT
      BAD = ( (NTERTS(2).LT.1).OR.(NTERTS(2).GT.9999) )
      IF(BAD) CALL MDWARN(
- 'BAD DAYS SINCE LAUNCH FROM MDP ANNOTATION')
      IF(.NOT.BAD) IOUT = NTERTS(2)

```

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

03ANOT  
000

```

      BAD = ( (NERTS(2).LT.1).OR.(NERTS(2).GT.9999) )
      IF(BAD) CALL MDWARN(
-      'BAD DAYS SINCE LAUNCH FROM MDP HEADER')
      IF(.NOT.BAD) IOUT = NERTS(2)
      NERTS(2) = IOUT
C
C
C -- HOUR.MINUTE.TENS OF SECONDS RELATIVE TO OBSERVATION
C
      IOUT = MAXINT
      BAD = ( (NERTS(3).LT.0).OR.(NERTS(3).GT.24000) )
      IF(BAD) CALL MDWARN(
-      'BAD GHT FROM MDP ANNOTATION')
      IF(.NOT.BAD) IOUT = NERTS(3)
      BAD = ( (NERTS(3).LT.0).OR.(NERTS(3).GT.24000) )
      IF(BAD) CALL MDWARN(
-      'BAD GHT FROM MDP HEADER')
      IF(.NOT.BAD) IOUT = NERTS(3)
      NERTS(3) = IOUT
C
      DO 310 I = 1,3
      IF(NERTS(I).EQ.MAXINT)CALL MDPATL(
-      'INVALID ERTS SCENE IDENTIFICATION')
      IF(NERTS(I).EQ.MAXINT) GO TO 320
310 CONTINUE
C
C
C DUMP BUFFER IF TRACE IS ON OR DIAGNOSTIC COUNT IS CHANGED
C
320 IF( (MTRACE.EQ.0).AND.(NDSAVE.EQ.NOTOTL) ) GO TO 323
      KTABLE(1) = '??????'      & MASK OUT NON-CHAR REC &. FILL. REC TYPE
      DO 322 LOWCHR = 1,ANOTSZ,48
      LOWINT = NI4NC(LOWCHR)
      CALL ERPRNT(1,NI4NC(48), KTABLE(LOWINT))
322 CONTINUE
C
C
C DETERMINE # OF REC TO BE READ TO POSITION BEFORE IMAGE REC
C
323 NUMRDS = 2
      IF( (LU3REF(1).NE.'AM').AND.(LU3REF(1).NE.'AR') ) NUMRDS = 1
C
C
C READ PAST EOF
C
      DO 325 NREADS = 1,NUMRDS
      IOADDR(LU3PKT) = LOC(KTABLE)
      IOSIZE(LU3PKT) = 36
      IOWAIT(LU3PKT) = 10
      IOFUNC(LU3PKT) = 'BK'      & READ FORWARD
      CALL ERION(LU3PKT)
325 CONTINUE
C
      IF(ICODE(LU3PKT).NE.'EOF') CALL MDPATL(
-      ' EOF NOT FOUND BETWEEN ANOT & IMAGE RECORDS')
C

```



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

03ANOT  
007

```
C
C CALL APPROPRIATE ROUTINE FOR SCENE SIZE AND INPUT WINDOW
C
  IF(LUSREF(1).EQ.'AR') CALL 03SZAR
  IF(LUSREF(1).EQ.'AM') CALL 03SZAM
  IF(LUSREF(1).EQ.'PM') CALL 03SZPM
  IF(LUSREF(1).EQ.'PR') CALL 03SZPR
C
  900 RETURN
C
  END
```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

03H0R  
001

SUBROUTINE 03H0R1    8 READ HEADER RECORD FROM MDP TAPES  
0 NOANCL    8 NUMBER OF ANCILLARY RECORDS

```

C
C
C
C HISTORY
C -----
C
C      J C CRISP      LEC      07/07/78      REQUIREMENTS
C      MARY TOMPKINS  LEC      10/28/78      ALGORITHM DESIGN
C      MARY TOMPKINS  LEC      11/9/78       ALGORITHM CODING
C
C
C METHOD
C -----
C
C      DEFINE HDRSZ PARAMETER.  SET NOANCL TO 0.  IF KTBLSZ<HDRSZ GENERATE
C      FATAL ERROR AND RETURN.  READ HDRSZ WORDS OF HEADER RECORD
C      INTO KTABLE.  IF BYTE FORMAT IS 'BB' CALL BST4BB.
C      CHECK BYTE 8 TO VERIFY HEADER RECORD -- IF NOT HEADER
C      RECORD, ISSUE FATAL ERROR.  EXTRACT BINARY INFORMATION AND SET
C      OF ANCILLARY RECORDS.  CALL CST4AS TO CONVERT HEADER RECORD TO
C      CHARACTER STRING.  EXTRACT/DECODE CHARACTER STRING INFORMATION
C      SET; LU3SEQ, LU3HIB, LU3BIB.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C
C      ERPRNT      8 PRINT IMAGE ON TTY OR LINE PRINTER (CHARACTER)
C      MDPATL      8 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C      MDPNOTE      8 PRINT/LOG 'NOTE' MESSAGES
C      BST4BB      8 INTERNAL BYTE STRING FOR 8-BIT EXTERNAL BYTE STRING
C      GETCHR      8 GET CHARACTER FROM CHARACTER STRING
C      MOVCHR      8 MOVE ONE CHAR. FROM ONE CHAR. STRING TO ANOTHER
C      CST4AS      8 CHARACTER STRING FOR ASCII BYTE STRING
C      GETBYT      8 GET NON-NEGATIVE INTEGER FROM BYTE IN BYTE STRING
C      ER10W      8 INITIATE I/O AND WAIT FOR COMPLETION
C      DCODE       8 DECODE NUMERIC CHARACTER STRING
C      MDPNOTE      8 PRINT/LOG 'NOTE' MESSAGES
C      DOUBLE PRECISION COS4CS    8 VARIABLE LENGTH CHAR STRING FOR FIXED LENGTH
C      INTEGER N04NI    8 # OF BYTES FOR # OR INTEGERS
C      INTEGER N04NI    8 # OF CHAR FOR # OF INTEGERS
C
C
C EXCEPTIONS
C -----
C
C
C      1. BEFORE THIS SUBROUTINE IS CALLED THE TAPE MUST BE POSITIONED AT
C      BEGINNING OF THE HEADER RECORD.
C

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

03HDR  
002

```

C      2. THE FOLLOWING CONDITIONS GENERATE THE DIAGNOSTICS SHOWN:
C
C      INVALID ENCODED ERTS NUMBER      NOTE ERROR
C      'EOF' I/O STATUS                  FATAL ERROR
C      'BADF' I/O STATUS                  FATAL ERROR
C      'LOST' I/O STATUS                  NOTE ERROR
C      'BAOR' I/O STATUS                  NOTE ERROR
C      KTBLSZ < MORSZ                    FATAL ERROR
C      RECORD TYPE NOT HEADER            FATAL ERROR
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST      % COMMON PROGRAM EXECUTION SWITCHES. COUNTERS
C      INCLUDE KOMTBL.LIST      % COMMON MULTI-PURPOSE TABLE
C      INCLUDE KOMNER.LIST      % ERTS SCENE PARAMETERS
C      INCLUDE KOMIO .LIST      % FORTRAN MANIPULATION OF ASH I/O PKTS.
C      INCLUDE KOMLU3.LIST      % PACKET/POINTERS FOR UNIT 3
C      INCLUDE MAXINT.LIST      % MAX INTEGER CONTAINED IN MACHINE
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER 0022/0022/ % RECORD TYPE FOR HEADER RECORD
C      INTEGER 18DAVL % BANDS AVAILABLE (BITS FOR BANDS / / / /4/5/6/7/8/
C      INTEGER NRECTY % RECORD TYPE
C      INTEGER INLEVE % TYPE OF DATE INTERLEAVING 'BSQ' OR 'BIL'
C      INTEGER NOSAVE % TEMP TO STORE % OF PREVIOUS DIAGNOSTICS
C      PARAMETER MORSZ = 899
C
C
C PROCEDURE
C -----
C
C      CALL TRACE
C
C      KTBLY = ' ' % FLAG PREVIOUS CONTENTS OF KTABLE AS DESTROYED
C      NOANCL = 0
C
C
C CHECK TO SEE IF KTABLE IS LARGE ENOUGH
C
C      IF(KTBLSZ.LT.MORSZ) CALL MCFATL( 'KTBLSZ < MORSZ')
C      IF(KTBLSZ.LT.MORSZ) GO TO 900
C
C
C READ MOP HEADER RECORD INTO KTABLE
C
C      IOADDR(LU3PKT) = LOC(KTABLE)
C      IOSIZE(LU3PKT) = MORSZ
C      IOUNIT(LU3PKT) = 10
C      IOFUNC(LU3PKT) = 'BK' % READ FORWARD
C      CALL ERION(LU3PKT)
C      ISTAT = IOCODE(LU3PKT)

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

03H0R  
003

```

      IF( (ISTAT.EQ.'EOF').OR.(ISTAT.EQ.'BADF') )CALL M0FATL(
      - C0B4CS(ISTAT,1,4). WHILE READING M0P HEADER RECORD')
C
      IF( (ISTAT.EQ.'BADR').OR.(ISTAT.EQ.'LOST') ) CALL M0NOTE(
      - C0B4CS(ISTAT,1,4). WHILE READING M0P HEADER RECORD')
C
C
C IF HEADER IS IN BB FORMAT. THEN CONVERT IT TO BST
C
      IF(LU3DPH.EQ.'BB') CALL BST4BB(KTABLE, KTABLE.N04NI(M0RSZ))
C
      CALL GETBYT(INRECTY, KTABLE,0) & OBTAIN RECORD TYPE CODE
      IF(INRECTY.NE.0022) CALL M0FATL(' RECORD TYPE OTHER THAN HEADER')
      IF(INRECTY.NE.0022) GO TO 900
C
C
C DETERMINE NUMBER OF ANCILLARY RECORDS
C
      CALL GET0BY(INOANCL, KTABLE,109)
C
C
C DETERMINE INTERLEAVING TYPE INDICATOR
C
      LU3SEQ(1) = 'BIL'
      LU3SEQ(2) = '0'
      CALL GETBYT(INLEVE, KTABLE,120)
      IF(INLEVE.EQ.0)LU3SEQ(1) = 'BSQ'
      IF(LU3SEQ(1).EQ.'BIL') CALL GETBYT(18DAVL, KTABLE,3500) & BINARY
C
C
C CONVERT ASCII TO CHARACTER INFORMATION
C
      CALL CST4AS(KTABLE, KTABLE.N04NI(M0RSZ))
C
C
C EXTRACT CHARACTER INFORMATION
C
      CALL GETCHR(INERADN, KTABLE,20) & 'A' OR 'D'
      CALL NOVST(INERPAT,1,3, KTABLE,21,3,' ') & NOMINAL PATH
      CALL NOVST(INERROW,1,3, KTABLE,24,3,' ') & NOMINAL ROW
C
      CALL DCODE(INERTS(1),REAL,K00TYP, KTABLE,9,1) & LANDSAT MISSION &
      IF(K00TYP.NE.'IN')INERTS(1) = MAXINT
C
      CALL DCODE(INERTS(2),REAL,K00TYP, KTABLE,9,4) & DAYS/LAUNCH
      IF(K00TYP.NE.'IN')INERTS(2) = MAXINT
C
      CALL DCODE(INERTS(3),REAL,K00TYP, KTABLE,13,5) & ONT
      IF(K00TYP.NE.'IN')INERTS(3) = MAXINT
C
      NERCHA = 0
      IF(INERTS(1).NE.3) NERCHA = 4
C
C
C DETERMINE BAND CHARACTERISTICS FOR BIL ONLY!!
C

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

03NDR  
004

```

      IF(LU3SEQ(1).NE.'01L'100 TO 000
      DO 900 I = 0,1,-1
        IF(MOD(10DAVL,2).NE.0)CALL CST4IN(NERBAN(1),1,NC4NI(1), 1+3,1)
        10DAVL = 10DAVL / 2
        CALL GETCHR(NERBAI(1), KTABLE,3500+1) & GAIN
        CALL GETCHR(NERTHO(1), KTABLE,3501+1) & TRANSMISSION MODE
      900 CONTINUE
C
      000 LU3M10 = KTBLSZ
      LU3B10 = NC4NI(KTBLSZ)
C
C
C PRINT REC IF TRACE IS ON OR DIAGNOSTIC COUNT IS CHANGED
C
      IF( (TRACE.EQ.0).AND.(NDSAVE.EQ.NOTOTL) ) GO TO 900
C
      DO 090 LOWCHR = 1,MORSZ,40
        LOWINT = NI4NC(LOWCHR)
        CALL ERPRNT(1,NI4NC(40), KTABLE(LOWINT))
      090 CONTINUE
C
C
      900 RETURN
C
      END

```

**BAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**O3SZAM  
001**

**SUBROUTINE O3SZAM 8 OBTAIN SCENE SIZE AND INPUT WINDOW FOR NBP AN TAPES**

```

C
C
C
C HISTORY
C -----
C
C      MARY TOMPKINS      LEC      11/00/70      ORI CODE IN O3ANOT
C      MAR. TOMPKINS      LEMSCO 01/21/00      SEPARATE INTO O3SZAM
C
C
C METHOD
C -----
C
C      DEFINE VIDEOSZ PARAMETER.  IF KIBLSZ < VIDEOSZ GENERATE FATAL ERROR
C      AND RETURN.  READ VIDEOSZ WORDS INTO KTABLE.  CHECK RECORD TYPE TO
C      VERIFY THAT RECORD IS IMAGE.  IF NOT CONTINUE TO READ TIL RECORD
C      IS AN IMAGE OR LU3CBR => LU3HBR.  IF LU3CBR => LU3HBR GENERATE
C      FATAL ERROR AND RETURN.  ESTABLISH PIXEL COUNT.  SET NERSAM.  MERLIN
C      AND INITIALIZE INPUT WINDOW PACKET.  REPOSITION TAPE TO INTER-RECORD
C      OAP BEFORE 1ST IMAGE RECORD.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      NOFATL      8 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C      ERPRNT      8 PRINT IMAGE ON TTY OR LINE PRINTER (CHARACTER)
C      GETBYT      8 GET ONE NON-NEG INTEGER FROM BYTE STRING
C      ERION       8 INITIATE I/O AND WAIT FOR COMPLETION
C      GET400      8 INTERNAL BYTE STRING FOR 8-BIT EXTERNAL BYTE STRING
C      ERTHAT      8 TIMED WAIT
C      HONOTE      8 PRINT/LOG 'NOTE' MESSAGES
C      DOUBLE PRECISION CDS4CS 8 VARIABLE LENGTH CHAR STRING FOR FIXED LENGTH
C      INTEGER NI4NC      8 4 OF INTEGERS FOR 4 CHARACTERS
C
C
C EXCEPTIONS
C -----
C
C      1. THE FOLLOWING CONDITIONS GENERATE THE DIAGNOSTICS SHOWN:
C
C      NERSAM OUTSIDE VALID RANGE      FATAL ERROR
C      KIBLSZ < VIDEOSZ                  FATAL ERROR
C      'EOF' I/O STATUS                  FATAL ERROR
C      'BADP' I/O STATUS                  FATAL ERROR
C      'LOST' I/O STATUS                  NOTE
C      'BADR' I/O STATUS                  NOTE
C
C
C GLOBAL DECLARATIONS

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

03SZAM  
002

C -----

C

INCLUDE KOMXQT.LIST	% COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
INCLUDE KOMLU3.LIST	% PACKET/POINTERS FOR UNITS
INCLUDE KOMNER.LIST	% ERTS SCENE PARAMETERS
INCLUDE KOMTBL.LIST	% COMMON MULTI-PURPOSE TABLE
INCLUDE KOMIO .LIST	% FORTRAN ALLOWING USE OF ASSEMBLER I/O PACKETS
INCLUDE MINDEF.LIST	% DEFINE STRUCTURE OF WINDOW PACKET
INCLUDE KOMIHW.LIST	% COMMON INPUT WINDOW PACKETS

C

C

C LOCAL DECLARATIONS

C -----

C

INTEGER NOSAVE	% TEMP TO STORE # OF PREVIOUS DIAGNOSTICS
INTEGER 0355/0355/	% MNEMONIC FOR OCTAL355 (IMAGE REC TYPE)
INTEGER NRECTY	% RECORD TYPE
INTEGER IXL CNT	% PIXEL COUNT
INTEGER IXL TMP	% TEMP. USED TO COMPUTE IXL CNT
PARAMETER VIDOSZ = 892	

C

C

C PROCEDURE

C -----

C

C

LU3CBR = 0

C

C

C READ VIDOSZ WORDS OF IMAGE RECORDS INTO KTABLE

C

```
IF(KTBLSZ.LT.VIDOSZ) CALL MDFATL( 'IN 03SZAM KTBLSZ < VIDOSZ')
IF(KTBLSZ.LT.VIDOSZ) GO TO 900
NOSAVE = NDTOTL
```

C

```
330 IADDR(LU3PKT) = LOC(KTABLE)
IOSIZE(LU3PKT) = VIDOSZ
IOWAIT(LU3PKT) = 0
IOFUNC(LU3PKT) = '8K' % READ FORWARD
CALL ERLOW(LU3PKT)
ISTAT = ICODE(LU3PKT)
IF( (ISTAT.EQ.'EOF').OR.(ISTAT.EQ.'BADF') )CALL MDFATL(
= CBS4CS(ISTAT,1,4), ' WHILE READING MOP IMAGE RECORD')
```

C

```
IF( (ISTAT.EQ.'BADR').OR.(ISTAT.EQ.'LOST') )CALL MDNOTE(
= CBS4CS(ISTAT,1,4), ' WHILE READING MOP IMAGE RECORD')
```

C

C

C IF IMAGE IS IN BB FORMAT, THEN CONVERT IT TO BST

C

```
IF(LU3BFM.EQ.'BB') CALL BST4BB(KTABLE, KTABLE.NB4NI(VIDOSZ))
```

C

```
CALL GETBYT(NRECTY, KTABLE,6) % RECORD TYPE CODE
IF(NRECTY.NE.0355)LU3CBR = LU3CBR + 1
IF( (NRECTY.NE.0355).AND.(LU3CBR.LT.LU3HBR) ) GO TO 330
IF(NRECTY.NE.0355) CALL MDFATL(
```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

0382AM  
003

```

      = 'RECORD NOT THE EXPECTED IMAGE RECORD')
      IF(NRECTY.NE.0388) GO TO 900
C
C
C ESTABLISH NERLIN.NERSAM.INPUT WINDOW.SCENE CENTER
C
      CALL GETBYT(IXLCNT, KTABLE.3588) & PIXEL CNT/SCAN LINE
      IXLTHP = IXLCNT*2**6
      CALL GETBYT(IXLCNT,KTABLE.3588)
      IXLCNT = IXLCNT + IXLTHP
C
      NERLIN = 2400
      NERSAM = IXLCNT + 6 & PIXEL OFFSET BET. BANDS
      IF(NERSAM.LT.3000.OR.NERSAM.GT.3500) CALL HDFATL(
      = 'INVALID PIXEL CNT/SCAN LINE FROM IMAGE RECORD')
C
      HSAIHW(HSAM.WMIN) = 1
      HSAIHW(HSAM.WMAX) = NERSAM
C
      HSAIHW(HLIN.WMIN) = 1
      HSAIHW(HLIN.WMAX) = NERLIN
C
      CTRLIN = 0.5*(NERLIN + 1)
      CTRSAM = 0.5*(NERSAM + 1)
C
C
C PRINT REC IF TRACE IS ON OR DIAGNOSTIC COUNT IS CHANGED
C
      IF( (MTRACE.EQ.0).AND.(NOSAVE.EQ.NOTOTL) ) GO TO 800
C
      DO 700 LOWCHR = 1.48.48
          LOWINT = NI4NC(LOWCHR)
          CALL ERPRNT(1.NI4NC(48), KTABLE(LOWINT))
700  CONTINUE
C
C
C REPOSITION TAPE TO INTER-RECORD GAP BEFORE 1ST IMAGE RECORD
C
      800 DO 810 I = 1.LU3CBR
          IOADDR(LU3PKT) = LOC(KTABLE)
          IOSIZE(LU3PKT) = 38
          IOWAIT(LU3PKT) = 0
          IOFUNC(LU3PKT) = 'BL' & READ BACKWARD
          CALL ERTWAT(2000) & REVERSE GENTLY
          CALL ERIOW(LU3PKT)
810  CONTINUE
C
      900  RETURN
C
      END

```



DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

03SZAR  
001

SUBROUTINE 03SZAR    : OBTAIN SCENE SIZE AND INPUT WINDOW FOR MDP AR TAPES

```

C
C
C
C HISTORY
C -----
C
C   MARY TOMPKINS      LEC   11/09/79      ORI CODE IN 03ANOT
C   MARY TOMPKINS      LENS CO 01/21/80     SEPARATE INTO 03SZAR
C
C
C METHOD
C -----
C
C   SET NERSAM AND NERLIN TO GODDARD NOMINAL VALUES. INITIALIZE
C   INPUT WINDOW PACKETS.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C   NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C   NONE.
C
C
C EXCEPTIONS
C -----
C
C   NONE.
C
C
C GLOBAL DECLARATIONS
C -----
C
C   INCLUDE KOMXQT.LIST      : COMMON PROGRAM EXECUTION SWITCHES. COUNTERS
C   INCLUDE KOMNER.LIST      : ERTS SCENE PARAMETERS
C   INCLUDE WINDEF.LIST      : DEFINE STRUCTURE OF WINDOW PACKET
C   INCLUDE KOMIHW.LIST      : COMMON INPUT WINDOW PACKETS .
C
C
C LOCAL DECLARATIONS
C -----
C
C   NONE.
C
C
C PROCEDURE
C -----
C
C   SET NUMBER OF LINES/SAMPLES.SCENE CENTER FOR FULL SCENE

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

0382AR  
002

```
C
  NERLIN = 4125
  NERSAN = 5375
  CTRLIN = 0.5*(NERLIN + 1)
  CTRSAN = 0.5*(NERSAN + 1)

C
C
C SET INPUT WINDOW PACKET
C
  HSAIWH(WSAM.WMIN) = 1
  HSAIWH(WSAM.WMAX) = NERSAN
  HSAIWH(WLIN.WMIN) = 1
  HSAIWH(WLIN.WMAX) = NERLIN

C
C
C 900 RETURN
C
  END
```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

03SZPM  
001

SUBROUTINE 03SZPM 8 OBTAIN SCENE SIZE AND INPUT WINDOW FOR NDP PH TAPES

```

C
C
C
C HISTORY
C -----
C
C      MARY TOMPKINS      LEC      11/09/79      ORI CODE IN 03ANOT
C      MARY TOMPKINS      LEMSCO 01/21/80      SEPARATE INTO 03SZPM
C
C
C METHOD
C -----
C
C      DEFINE VIDEOSZ PARAMETER.  IF KTBLSZ < VIDEOSZ GENERATE FATAL ERROR
C      AND RETURN.  READ VIDEOSZ WORDS INTO KTABLE.  CHECK RECORD TYPE TO
C      VERIFY THAT RECORD IS IMAGE.  IF NOT CONTINUE TO READ TIL RECORD
C      IS AN IMAGE OR LU3CBR => LU3MBR.  IF LU3CBR => LU3MBR GENERATE
C      FATAL ERROR AND RETURN.  ESTABLISH FILL COUNT.  SET NERSAM, NERLIN
C      AND INITIALIZE INPUT WINDOW PACKET.  REPOSITION TAPE TO INTER-RECORD
C      GAP BEFORE 1ST IMAGE RECORD.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      MDFATL      8 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C      ERPRNT      8 PRINT IMAGE ON TTY OR LINE PRINTER (CHARACTER)
C      ERLOW      8 INITIATE I/O AND WAIT FOR COMPLETION
C      BST488      8 INTERNAL BYTE STRING FOR 8-BIT EXTERNAL BYTE STRING
C      GETBYT      8 GET ONE NON-NEG INTEGER FROM BYTE STRING
C      GETNYB      8 GET NON-NEG INTEGER FROM NYBLE IN BYTE STRING
C      ERTHAT      8 TIMED WAIT
C      MDNOTE      8 PRINT/LOG 'NOTE' MESSAGES
C      DOUBLE PRECISION C8S4CS 8 VAR LENGTH CHAR STRING FOR FIXED LENGTH CHAR
C      INTEGER N14NC      8 # OF INTEGERS FOR # CHARACTERS
C
C
C EXCEPTIONS
C -----
C
C      1. THE FOLLOWING CONDITIONS GENERATE THE DIAGNOSTICS SHOWN:
C
C      NERSAM OUTSIDE VALID RANGE      FATAL ERROR
C      KTBLSZ<VIDOSZ                    FATAL ERROR
C      'EOF' I/O STATUS                  FATAL ERROR
C      'BADF' I/O STATUS                  FATAL ERROR
C      'LOST' I/O STATUS                  NOTE
C      'BADR' I/O STATUS                  NOTE
C
C

```

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

03SZPM  
002

C GLOBAL DECLARATIONS

C -----  
C

INCLUDE KONXQT.LIST	% COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
INCLUDE KOHLU3.LIST	% PACKET/POINTERS FOR UNIT3
INCLUDE KOHNER.LIST	% ERTS SCENE PARAMETERS
INCLUDE KOMTBL.LIST	% COMMON MULTI-PURPOSE TABLE
INCLUDE KOMIO .LIST	% FORTRAN ALLOWING USE OF ASSEMBLER I/O PACKETS
INCLUDE MINDEF.LIST	% DEFINE STRUCTURE OF WINDOW PACKET
INCLUDE KOMINH.LIST	% COMMON INPUT WINDOW PACKETS

C

C

C LOCAL DECLARATIONS

C -----  
C

INTEGER NOSAVE	% TEMP TO STORE # OF PREVIOUS DIAGNOSTICS
INTEGER 0355/0355/	% MNEMONIC FOR OCTAL355 (IMAGE REC TYP)
INTEGER NRECTY	% RECORD TYPE
INTEGER NLFTFL	% LEFT FILL COUNT
INTEGER NRHTFL	% RIGHT FILL COUNT
INTEGER NTMPFL	% TEMP. TO COMPUTE RIGHT AND LEFT FILL CNTS.
INTEGER MINFCT	% MIN PIXEL FILL COUNT
INTEGER NOMSAM	% GOODARD'S NOMINAL NUMBER OF SAMPLES
PARAMETER VIDEOSZ = 892	

C

C

C PROCEDURE

C -----  
C

C

C

LU3CBR = 0

C

C

C READ VIDEOSZ WORDS OF IMAGE RECORDS INTO KTABLE

C

IF(KTBLSZ.LT.VIDEOSZ) CALL M0FATL( 'IN 03SZPM KTBLSZ < VIDEOSZ')  
IF(KTBLSZ.LT.VIDEOSZ) GO TO 900  
NOSAVE = NDTOTL

C

330 IOADDR(LU3PKT) = LOC(KTABLE)  
IOSIZE(LU3PKT) = VIDEOSZ  
IOWAIT(LU3PKT) = 0  
IOFUNC(LU3PKT) = 'RK' % READ FORWARD  
CALL ER10W(LU3PKT)  
ISTAT = ICODE(LU3PKT)  
IF( (ISTAT.EQ.'EOF').OR.(ISTAT.EQ.'BADF') )CALL M0FATL(  
= CBS4CS(ISTAT,1,4), ' WHILE READING MDP IMAGE RECORD')

C

IF( (ISTAT.EQ.'BADR').OR.(ISTAT.EQ.'LOST') ) CALL M0NOTE(  
= CBS4CS(ISTAT,1,4), ' WHILE READING MDP IMAGE RECORD')

C

C

C IF IMAGE IS IN 88 FORMAT, THEN CONVERT IT TO 85T

C

IF(LU3BFH.EQ.'88') CALL 85T488(KTABLE, KTABLE.NB4NI(VIDEOSZ))

C

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

038ZPM  
003

```

CALL GETBYT(INRECTY, KTABLE,6) & RECORD TYPE CODE
IF(INRECTY.NE.0355)LU3CBR = LU3CBR + 1
IFI (INRECTY.NE.0355).AND.(LU3CBR.LT.LU3HBR) ) GO TO 330
IF(INRECTY.NE.0355) CALL MOFATL(
- 'RECORD NOT THE EXPECTED IMAGE RECORD')
IF(INRECTY.NE.0355) GO TO 900

C
C
C ESTABLISH LEFT & RIGHT FILL COUNTS ON 'PH' TAPES
C
C -- LEFT FILL COUNT
C
CALL GETBYT(NLFTFL, KTABLE,10)
CALL GETNYB(NTMPFL, KTABLE,11,1)
NLFTFL = NLFTFL*2**4 + NTMPFL

C
C -- RIGHT FILL COUNT
CALL GETNYB(NRHTFL, KTABLE,11,2)
CALL GETBYT(NTMPFL, KTABLE,12)
NRHTFL = NRHTFL*2**8 + NTMPFL

C
C ESTABLISH NERLIN,NERSAM,INPUT WINDOW,SCENE CENTER
C
NERLIN = 2983
NOMSAM = 3548

C
MINFCT = MAX0(MIN0(NLFTFL,NRHTFL)-8,0) & MIN PIXEL FILL COUNT
MSAIHW(HSAM,WMIN) = MINFCT + 1
MSAIHW(HSAM,WMAX) = NOMSAM - MINFCT

C
MSAIHW(WLIN,WMIN) = 1
MSAIHW(WLIN,WMAX) = NERLIN

C
NERSAM = MSAIHW(HSAM,WMAX) - MSAIHW(HSAM,WMIN) + 1
IF(NERSAM.LT.3350.OR.NERSAM.GT.NOMSAM) CALL MOFATL(
- 'BAD SAMPLE NUMBER COMPUTED FROM IMAGE RECORD')

C
CTRLIN = 0.5*(NERLIN + 1)
CTRSAM = 0.5*(NERSAM + 1) + MINFCT

C
C PRINT REC IF TRACE IS ON OR DIAGNOSTIC COUNT IS CHANGED
C
IFI (INTRACE.EQ.0).AND.(NDSAVE.EQ.NOTOTL) ) GO TO 800

C
DO 700 LOWCHR = 1,48,48
LOWINT = NI4NC(LOWCHR)
CALL ERPRNT(1,NI4NC(48), KTABLE(LOWINT))
700 CONTINUE

C
C REPOSITION TAPE TO INTER-RECORD GAP BEFORE 1ST IMAGE RECORD
C
800 DO 810 I = 1,LU3CBR
IOADDR(LU3PKT) = LOC(KTABLE)

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**038ZPH  
004**

```
      IOSIZE(LU3PKT) = 36  
      IOWAIT(LU3PKT) = 0  
      IOFUNC(LU3PKT) = 'RL'  & READ BACKWARD  
      CALL ERTNAT(2000)  & REVERSE GENTLY  
      CALL ERION(LU3PKT)  
C 010 CONTINUE  
C  
C 900 RETURN  
C  
      END
```

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

03SZPR  
001

SUBROUTINE 03SZPR 8 OBTAIN SCENE SIZE AND INPUT WINDOW FOR MDP PR TAPES

```

C
C
C
C HISTORY
C -----
C
C      MARY TOMPKINS      LEC      11/09/79      ORIGINAL CODE IN 03ANOT
C      MARY TOMPKINS      LEMSCO 01/21/80      SEPARATE INTO 03SZPR
C
C
C METHOD
C -----
C
C      DEFINE VIDEOSZ PARAMETER.  IF KTBLSZ < VIDEOSZ GENERATE FATAL ERROR
C      AND RETURN.  READ VIDEOSZ WORDS INTO KTABLE.  CHECK RECORD TYPE TO
C      VERIFY THAT RECORD IS IMAGE.  IF NOT CONTINUE TO READ TIL RECORD
C      IS AN IMAGE OR LU3CBR => LU3MBR.  IF LU3CBR => LU3MBR GENERATE
C      FATAL ERROR AND RETURN.  ESTABLISH FILL COUNT.  SET NERSAM, NERLIN
C      AND INITIALIZE INPUT WINDOW PACKET.  REPOSITION TAPE TO INTER-RECORD
C      GAP BEFORE 1ST IMAGE RECORD.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      MDPATL      8 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C      ERPRNT      8 PRINT IMAGE ON TTY OR LINE PRINTER (CHARACTER)
C      ERIOH       8 INITIATE I/O AND WAIT FOR COMPLETION
C      BST400      8 INTERNAL BYTE STRING FOR 8-BIT EXTERNAL BYTE STRING
C      GETBYT      8 GET ONE NON-NEG INTEGER FROM BYTE STRING
C      GETDBY      8 GET NON-NEG INTEGER FROM DOUBLE-BYTE IN BYTE STRING
C      ERTWAT      8 TIMED WAIT
C      MDNOTE      8 PRINT/LOG 'NOTE' MESSAGES
C      DOUBLE PRECISION CBS4CS 8 VAR LENGTH CHAR STRING FOR FIXED LENGTH CHAR
C      INTEGER NI4NC      8 4 OF INTEGERS FOR 4 CHARACTERS
C
C
C EXCEPTIONS
C -----
C
C      1. THE FOLLOWING CONDITIONS GENERATE THE DIAGNOSTICS SHOWN:
C          NERSAM OUTSIDE VALID RANGE      FATAL ERROR
C          KTBLSZ<VIDOSZ                    FATAL ERROR
C          'EOF' I/O STATUS                  FATAL ERROR
C          'BADF' I/O STATUS                 FATAL ERROR
C          'LOST' I/O STATUS                 NOTE
C          'BADR' I/O STATUS                 NOTE

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

03SZPH  
002

C GLOBAL DECLARATIONS

C -----  
C

INCLUDE KONXQT.LIST	% COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
INCLUDE KONLU3.LIST	% COMMON PACKET/POINTERS FOR UNIT 3
INCLUDE KONNER.LIST	% COMMON ERTS SCENE PARAMETERS
INCLUDE KONBL.LIST	% COMMON MULTI-PURPOSE TABLE
INCLUDE KONIO .LIST	% FORTRAN ALLOWING USE OF ASSEMBLER I/O PACKETS
INCLUDE WINDEF.LIST	% DEFINE STRUCTURE OF WINDOW PACKET
INCLUDE KONIMW.LIST	% COMMON INPUT WINDOW PACKETS

C

C

C LOCAL DECLARATIONS

C -----  
C

INTEGER NDSAVE	% TEMP TO STORE # OF PREVIOUS DIAGNOSTICS
INTEGER 0355/0355/	% MNEMONIC FOR OCTAL355 (IMAGE REC TYP)
INTEGER NRECTY	% RECORD TYPE
INTEGER NLFTFL	% LEFT FILL COUNT
INTEGER NRHTFL	% RIGHT FILL COUNT
INTEGER MINFCT	% MIN PIXEL FILL COUNT
INTEGER NONSAM	% GOODARD'S NOMINAL NUMBER OF SAMPLES
PARAMETER VIDOSZ = 892	

C

C

C PROCEDURE

C -----  
C

C

C

LU3CBR = 0

C

C

C READ VIDOSZ WORDS OF IMAGE RECORDS INTO KTABLE

C

```
IF(KTBLSZ.LT.VIDOSZ) CALL MDPATL( 'IN 03SZPH KTBLSZ < VIDOSZ')
IF(KTBLSZ.LT.VIDOSZ) GO TO 900
NDSAVE = NDTOTL
```

C

```
330 I0ADDR(LU3PKT) = LOC(KTABLE)
I0SIZE(LU3PKT) = VIDOSZ
I0WAIT(LU3PKT) = 0
I0FUNC(LU3PKT) = '8K' % READ FORWARD
CALL ER10W(LU3PKT)
ISTAT = I0CODE(LU3PKT)
IF( (ISTAT.EQ.'EOF').OR.(ISTAT.EQ.'BADF') )CALL MDPATL(
= C054CS(ISTAT.1.4),' WHILE READING MDP IMAGE RECORD')
```

C

```
IF( (ISTAT.EQ.'BADR').OR.(ISTAT.EQ.'LOST') ) CALL MDPNOTE(
= C054CS(ISTAT.1.4),' WHILE READING MDP IMAGE RECORD')
```

C

C

C IF IMAGE IS IN BB FORMAT, THEN CONVERT IT TO BT

C

```
IF(LU3BFH.EQ.'BB') CALL BT4BB(KTABLE, KTABLE.N04NI(VIDOSZ))
```

C

```
CALL GETBYT(NRECTY, KTABLE.0) % RECORD TYPE CODE
```



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

0382PR  
003

```

IF(INRECTY.NE.0355)LU3CBR = LU3CBR + 1
IF( INRECTY.NE.0355).AND.(LU3CBR.LT.LU3MBR) ) GO TO 330
IF(INRECTY.NE.0355) CALL M0FATL(
  - 'RECORD NOT THE EXPECTED IMAGE RECORD')
IF(INRECTY.NE.0355) GO TO 900

```

C  
C  
C  
C

ESTABLISH RIGHT LEFT FILL COUNTS FOR 'PR' TAPES

C  
C  
C

CALL GETDBY(NLFTFL, KTABLE,0) & LEFT FILL CNT.

CALL GETDBY(NRHTFL, KTABLE,1) & RIGHT FIL CNT.

C  
C  
C  
C

ESTABLISH VALUES FOR NERLIN.NERSAM.SCENE CENTER

C  
C  
C  
C  
C

NERLIN = 5322

NOMSAM = 5322

C  
C  
C  
C  
C

MINFCT = MAX0(MIND(NLFTFL,NRHTFL)-0.0) & MIN PIXEL FILL COUNT

MSA1HW(MSAM,WMIN) = MINFCT + 1

MSA1HW(MSAM,WMAX) = NOMSAM - MINFCT

C  
C  
C

MSA1HW(NLIN,WMIN) = 1

MSA1HW(NLIN,WMAX) = NERLIN

C  
C  
C  
C

NERSAM = MSA1HW(MSAM,WMAX) - MSA1HW(MSAM,WMIN) + 1

IF(NERSAM.LT.5000.OR.NERSAM.GT.NOMSAM) CALL M0FATL(

- 'BAD SAMPLE NUMBER COMPUTED FROM IMAGE RECORD')

C  
C  
C

CTRLIN = 0.5\*(NERLIN + 1)

CTRSAM = 0.5\*(NERSAM + 1) + MINFCT

C  
C  
C  
C

PRINT REC IF TRACE IS ON OR DIAONOSTIC COUNT IS CHANGED

C  
C  
C

IF( (TRACE.EQ.0).AND.(NDSAVE.EQ.NDTOTL) ) GO TO 800

C  
C  
C  
C

DO 700 LOWCHR = 1,48,48

LOWINT = N14NC(LOWCHR)

CALL ERPRNT(1,N14NC(48), KTABLE(LOWINT))

700 CONTINUE

C  
C  
C  
C

REPOSITION TAPE TO INTER-RECORD GAP BEFORE 1ST IMAGE RECORD

```

800 GO 810 1 = 1,LU3CBR
  IOADDR(LU3PKT) = LOC(KTABLE)
  IOSIZE(LU3PKT) = 36
  IOWAIT(LU3PKT) = 0
  IOFUNC(LU3PKT) = 'BL' & READ BACKWARD
  CALL ERTHAT(2000) & REVERSE GENTLY
  CALL ER10W(LU3PKT)
810 CONTINUE

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**0302PM  
004**

**C  
000 RETURN  
C  
END**

SUBROUTINE O3TOR 8 READ TAPE DIRECTORY RECORD FROM MDP TAPES

```

C
C
C
C HISTORY
C -----
C
C   CHARLES MELHKE   LEC   08/28/79   REQUIREMENTS
C   J C CRISP       LEC   09/21/79   ALGORITHM DESIGN
C   J C CRISP       LEC   09/24/79   ALGORITHM CODING
C
C
C METHOD
C -----
C
C   DEFINE TORSZ PARAMETER.  IF KTBSZ<TORSZ GENERATE FATAL ERROR AND RETURN.
C   INITIALIZE L1/VOL TO 1 AND L13VH1.  READ TORSZ WORDS OF TAPE DIRECTORY
C   RECORD INTO KTABLE.  IF L13BPM = '00' CALL 0ST400.  CHECK BYTE 8 TO VERIFY
C   TAPE DIRECTORY.  IF I/O ERRORS ISSUE DIAGNOSTIC.  ELSE IF NOT TAPE
C   DIRECTORY, ISSUE FATAL ERROR AND RETURN.  EXTRACT/DECODE VOLUME NUMBER
C   AND NUMBER OF VOLUMES.  IF VOLUME <> 1 ISSUE FATAL ERROR.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C   NONE
C
C
C EXTERNAL REFERENCES
C -----
C
C   ERPRNT      8 PRINT IMAGE ON TTY OR LINE PRINTER (CHARACTER)
C   ERTHAT      8 TIMED WAIT
C   ERION       8 INITIATE I/O AND WAIT FOR COMPLETION
C   0ST400      8 INTERNAL BYTE STRING FOR 8-BIT EXTERNAL BYTE STRING
C   M0FATL      8 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C   0ETBYT      8 GET NON-NEGATIVE INTEGER FROM BYTE IN BYTE STRING
C   CST4AS      8 CHARACTER STRING FOR ASCII BYTE STRING
C   MOVCHR      8 MOVE CHARACTER FROM ONE CHARACTER STRING TO ANOTHER
C   MOVCS      8 MOVE SUBSTRING FROM ONE CHARACTER STRING TO ANOTHER
C   DCODE       8 DECODE NUMERIC CHARACTER STRING
C   M0NOTE      8 PRINT/LOG 'NOTE' MESSAGES
C   DOUBLE PRECISION C0S4CS  8 VARIABLE LENGTH CHAR STRING FOR FIXED LENGTH
C   INTEGER M14NC  8 # OF INTEGERS FOR # OF CHARS
C
C
C EXCEPTIONS
C -----
C
C   1. THE FOLLOWING CONDITIONS GENERATE THE DIAGNOSTICS SHOWN
C
C       'EOF' I/O STATUS      FATAL ERROR
C       'BADP' I/O STATUS     FATAL ERROR
C       'LOST' I/O STATUS     NOTE
C       'BADR' I/O STATUS     NOTE
C       RECORD TYPE <> DIRECTORY  FATAL ERROR

```

**DAN PACKAGE APPENDIX H  
UTILITY ROUTINES**

**O3TOR  
002**

```

C          VOLUME NUMBER (>) 1          FATAL ERROR
C          KTBLSZ < TORSZ          FATAL ERROR
C          HIGH VOL NUMBER INVALID      FATAL ERROR
C
C
C GLOBAL DECLARATIONS
C -----
C
C          INCLUDE KONXQT.LIST          8 COMMON PROGRAM EXECUTION SWITCHES.COUNTERS
C          INCLUDE KONTBL.LIST          8 COMMON MULTI-PURPOSE TABLE
C          INCLUDE KONIO .LIST          8 FORTRAN MANIPULATION OF ASH I/O PACKETS
C          INCLUDE KONNER.LIST          8 COMMON ERTS SCENE PARAMETERS
C          INCLUDE KOHLUS.LIST          8 COMMON PACKET POINTERS FOR UNIT 3
C
C
C LOCAL DECLARATIONS
C -----
C
C          INTEGER O011/G311/          8 TAPE DIRECTORY CODE IN OCTAL
C          INTEGER I0YT                8 TAPE DIRECTORY CODE
C          INTEGER NDSAVE                8 TEMP TO STORE # OF PREVIOUS DIAGNOSTICS
C          PARAMETER TORSZ = 80
C
C
C PROCEDURE
C -----
C
C          CALL TRACE
C
C
C INITIALIZE AND CHECK TAPE DIRECTORY BUFFER
C
C          IF(TORSZ.GT.KTBLSZ) CALL M0FATL( 'TORSZ > KTBLSZ IN O3TOR')
C          IF(TORSZ.GT.KTBLSZ) GO TO 900
C          NDSAVE = NDTOTL
C
C
C READ TAPE DIRECTORY INTO KTABLE
C
C          I0ADDR(IU3PKT) = LOC(KTABLE)
C          I0SIZE(IU3PKT) = TORSZ
C          I0WAIT(IU3PKT) = 10
C          I0FUNC(IU3PKT) = '8K' 8 READ FORWARD
C          CALL ERTWAT(2000)
C          CALL ERTION(IU3PKT)
C          ISTAT = I0CODE(IU3PKT)
C          IF( (ISTAT.EQ.'EOF').OR.(ISTAT.EQ.'BADF') )CALL M0FATL(
C          = C0S4CS(ISTAT,1,4).' WHILE READING DIRECTORY RECORD')
C
C          IF( (ISTAT.EQ.'LOST').OR.(ISTAT.EQ.'BADR') )CALL M0NOTE(
C          = C0S4CS(ISTAT,1,4).' WHILE READING DIRECTORY RECORD')
C
C
C CHECK IF DATA IS IN INTERNAL FORMAT
C
C          IF(IU3DPH.EQ.'80') CALL 00T000(KTABLE, KTABLE.N04N(I(TORSZ))

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

03TOR  
003

```

C
C
C VERIFY TAPE DIRECTORY
C
    CALL GETBYT(1BYT ,KTABLE.6)
    IF(1BYT.NE.0011) CALL M0FATL( 'FIRST RECORD NOT TAPE DIRECTORY')
C
C
C CONVERT ALPHANUMERIC INFORMATION TO CHARACTER STRING FROM ASCII BYTE STRING
C
    CALL CST4AS(KTABLE. KTABLE.NB4NI(TORSZ))
C
C
C EXTRACT/DECODE INFORMATION
C
    CALL DCODE(LU3VHI.REAL.KODTYP. KTABLE.20.1) 3 TOTAL VOLS
    IF(LU3VHI.LT.1.OR.LU3VHI.GT.V3MAX) KODTYP = 'ERR'
    IF(KODTYP.NE.'IN') CALL M0FATL(
    = 'INVALID TAPE VOLUME UPPER LIMIT IN TAPE DIRECTORY:'. ' '
    & CBS4CS(KTABLE.(20).(1)))
    NCCT = 1
    NCCTOT = 1
C
    CALL DCODE(LU3VOL.REAL.KODTYP. KTABLE.19.1)
    IF(LU3VOL.NE.1) KODTYP = 'ERR'
    IF(KODTYP.NE.'IN') CALL M0FATL(
    = 'BAD TAPE VOLUME NUMBER IN TAPE DIRECTORY ON FIRST REEL:'. ' '
    & CBS4CS(KTABLE.(19).(1)) )
C
    CALL MOVCHR(LU3REF(1).1. KTABLE.11) 3 'P' OR 'A'
    CALL MOVCS(TU3REF(1).2.5. KTABLE.9.1.' ') 3 'M' OR 'R'
C
    LU3REF(2) = '0' 3 MDP 1978
C
C
C DUMP BUFFER IF TRACE IS ON OR DIAGNOSTIC COUNT IS CHANGED
C
    IF( (TRACE.EQ.0).AND.(NDSAVE.EQ.NOTOTL) ) GO TO 800
    CALL ERPRNT(1.1.'*TDR*')
    DO 700 LOWCHR = 1.TORSZ.48
        LOWINT = NI4NC(LOWCHR)
        CALL ERPRNT(1.NI4NC(48). KTABLE(LOWINT))
700    CONTINUE
C
C
C READ OVER EOF
C
800    IOFJNC(LU3PKT) = 'BK' 3 READ FORWARD
    CALL ERJOW(LU3PKT)
    ISTAT = ICODE(LU3PKT)
    IF(ISTAT.NE.'EOF') CALL M0FATL(
    = 'EXPECTED EOF AFTER TAPE DIRECTORY NOT FOUND')
C
900    RETURN
C
    END

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PITROL  
001

SUBROUTINE PITROL:   8 ESTIMATE PITCH AND ROLL  
1 CTRLAT,CTRLON.   8 SCENE CENTER (DEGREES)  
1 DIRLAT,DIRLON.   8 SCENE NA DIR (DEGREES)  
1 HMYDEG.           8 HEADING MINUS YAW (DEGREES)  
1 ALTKM.            8 ALTITUDE (KILOMETRES)  
0 PITDEG,ROLOEG)   8 PITCH & ROLL (DEGREES)  
-----

C  
C  
C  
C HISTORY  
C -----

C       E M SCHLOSSER       LEC       05/03/73       ORIGINAL CODE  
C       E M SCHLOSSER       LEMSCO   05/27/80       PITCH/ROLL UNDEFINED IF ABSVAL > 9  
C  
C

C METHOD  
C -----

C       TRANSFORM CENTER & NA DIR LAT/LONG TO TRANSVERSE MERCATOR EAST/NORTH.  
C       ESTIMATE PITCH & ROLL.

C       ERTS CONVENTIONS FOR ATTITUDE AND HEADING ARE AS FOLLOWS:  
C       POSITIVE PITCH IS NOSE DOWN  
C       POSITIVE ROLL IS CLOCKWISE VIEWED FROM BEHIND  
C       POSITIVE YAW IS COUNTERCLOCKWISE VIEWED FROM ABOVE  
C       POSITIVE HEADING IS CLOCKWISE VIEWED FROM ABOVE  
C  
C

C MACHINE-DEPENDENT CODE  
C -----

C       NONE.  
C  
C

C EXTERNAL REFERENCES  
C -----

C       U40               8 UTM (OR STM) COORDINATES FOR GEOGRAPHIC COORDINATES  
C  
C

C EXCEPTIONS  
C -----

C       1. IF NA DIR IS UNDEFINED OR COMPUTED PITCH OR ROLL HAS MAGNITUDE  
C       GREATER THAN 9 DEGREES, THEN PITCH AND ROLL ARE SET TO UNDEFINED.  
C  
C

C GLOBAL DECLARATIONS  
C -----

C       NONE.  
C  
C

C LOCAL DECLARATIONS  
C -----

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PITROL  
882

C  
REAL CENE,CENN                   8 CENTER EASTING, NORTHING  
REAL DIRE,DIRN                 8 NADIR EASTING, NORTHING  
REAL PITKH,ROLKH               8 CENTER/NADIR DISPLACEMENT ALONG PITCH, ROLL AXES

C  
C  
C PROCEDURE  
C -----  
C

```

CALL TRACE
IF((ABS(DIRLAT).GT.360.).AND.(ABS(DIRLON).GT.360.)) GO TO 800
  CALL U40(CENE,CENN,
    CTRLAT,CTRLON,DIRLON)
  CALL U40(DIRE,DIRN,
    DIRLAT,DIRLON,DIRLON)
  HMYRAD=(HMYDEG-180.)*.017453292
  PITKH=.001*COS(HMYRAD)*((CENN-DIRN)*(CENE-DIRE)*TAN(HMYRAD))
  ROLKH=.001*COS(HMYRAD)*((CENE-DIRE)*(DIRN-CENN)*TAN(HMYRAD))
  PITDEG=57.295780*ATAN(PITKH/ALTKH)
  ROLDEG=57.295780*ATAN(ROLKH/ALTKH)
  IF((ABS(PITDEG).LT.9.).AND.
    (ABS(ROLDEG).LT.9.)) GO TO 900
800 PITDEG=-9999.
   ROLDEG=-9999.
900 RETURN
END

```

**DAN PACKAGE APPENDIX H  
UTILITY ROUTINES**

**PRNUM  
001**

```

SUBROUTINE PRNUM( 8 PRINT BOX NUMBERS (VARIABLE HEIGHT)
.
I IUNIT. 8 OUTPUT UNIT NUMBER
I NLBODY. 8 NUMBER OF LINES IN BOX TYPE BODY (12 TO 66)
I NLFACE. 8 NUMBER OF LINES IN BOX TYPE FACE (12 TO 60)
I N1,N2,N3,N4,N5,N6,N7) 8 SEVEN 1-DIGIT INTEGERS TO BE PRINTED IN BOX TYPE
-----
C
C
C
C HISTORY
C -----
C
C E H SCHLOSSER LEC 01/05/74 ORIGINAL CODE IN PRNUM
C E H SCHLOSSER LEC 12/27/79 RENAME. CHANGE ARGOS. ELIMINATE 8ADD
C
C
C METHOD
C -----
C
C FOR EACH NUMBER, DETERMINE FOREGROUND & BACKGROUND CHARACTERS:
C CASE 1: ( 0 <= NUM <= 9) FOREGR := ENCODED NUM. BACKGR := ' '
C CASE 2: (10 <= NUM <= 19) FOREGR := ' '. BACKGR := ENCODED NUM
C CASE 3: (20 <= NUM ) FOREGR := ' '. BACKGR := ' '
C PRINT BACKGROUND CHARACTERS ON TYPE BODY LINES (IF ANY) ABOVE FACE.
C PRINT BOX DIGITS ON TYPE FACE LINES.
C PRINT BACKGROUND CHARACTERS ON TYPE BODY LINES (IF ANY) BELOW FACE.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C DIMENSION AND FORMAT SPECIFICATIONS ASSUME 6 CHARS PER INTEGER.
C
C
C EXTERNAL REFERENCES
C -----
C
C CST4IN 8 CHARACTER STRING FOR INTEGER
C MOVCS 8 MOVE CHARACTER STRING
C LBOX4I 8 LINE OF BOX DIGIT FOR INTEGER
C
C
C EXCEPTIONS
C -----
C
C 1. IF ANY OF THE SEVEN 1-DIGIT INTEGERS IS NEGATIVE. THEN RESULTS ARE
C UNDEFINED.
C
C 2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
C
C
C GLOBAL DECLARATIONS
C -----
C
C NONE.
C

```



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PRNUM  
002

```

C
C LOCAL DECLARATIONS
C -----
C
      INTEGER IDIGIT(7)      & 1-DIGIT INTEGERS
      INTEGER KHRFOR(7)      & FOREGROUND CHARACTER FOR EACH IDIGIT
      INTEGER KHRBAK(7)      & BACKGROUND CHARACTER FOR EACH IDIGIT
      INTEGER NBXD           & NUMBER (1 THRU 7) OF CURRENT BOX DIGIT
      INTEGER KASDIO         & CASE FOR CURRENT IDIGIT:
C                               1 --- 0 <= IDIGIT <= 9 (BLACK ON WHITE)
C                               2 --- 10 <= IDIGIT <= 19 (WHITE ON BLACK)
C                               3 --- 20 <= IDIGIT (BLANK)
      INTEGER NORDUP(12)     & ORDER FOR DUPLICATING LINES 1 THRU 12:
& /99.99.12.10. 8.99.99. 7. 9.11.99.99/
      INTEGER KMBUFO(28)     & OUTPUT CHARACTER BUFFER
      DEFINE KHLEFT(1)=KMBUFO(1*4-3) & COLUMNS OF TYPE BODY LEFT OF FACE
      DEFINE KHFACE(1)=KMBUFO(1*4-2) & COLUMNS OF TYPE FACE
      DEFINE KHRITE(1)=KMBUFO(1*4-0) & COLUMNS OF TYPE BODY RIGHT OF FACE
C
C
C PROCEDURE
C -----
C
      CALL TRACE
C
C
C INITIALIZE
C
      IDIGIT(1)=N1
      IDIGIT(2)=N2
      IDIGIT(3)=N3
      IDIGIT(4)=N4
      IDIGIT(5)=N5
      IDIGIT(6)=N6
      IDIGIT(7)=N7
C
C
C DETERMINE FOREGROUND & BACKGROUND CHARACTERS
C
      DO 180 NBXD=1,7
        IDIGIT(NBXD)=ABS(IDIGIT(NBXD))
        KASDIO=MING(IDIGIT(NBXD)/10.2)+1
        GO TO(110,120,130), KASDIO
110      CONTINUE      & 0 <= IDIGIT <= 9
        CALL CST4IN(KHRFOR(NBXD),(1),(1), IDIGIT(NBXD),1)
        KHRBAK(NBXD)=' '
        GO TO 180
120      CONTINUE      & 10 <= IDIGIT <= 19
        IDIGIT(NBXD)=IDIGIT(NBXD)-10
        KHRFOR(NBXD)=' '
        CALL CST4IN(KHRBAK(NBXD),(1),(1), IDIGIT(NBXD),1)
        GO TO 180
130      CONTINUE      & 20 <= IDIGIT
        KHRFOR(NBXD)=' '
        KHRBAK(NBXD)=' '
180 CONTINUE

```

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

PRNUM  
003

```

C
C
C INITIALIZE OUTPUT BUFFER TO BACKGROUND CHARACTERS
C
      DO 200 NBXD=1,7
        CALL MOVCST(KHLEFT(NBXD),(1),(21),
-          KHRBAK(NBXD),(1),(1),KHRBAK(NBXD))
      200 CONTINUE
C
C
C PRINT BODY LINES ABOVE FACE (IF ANY)
C
      NLMAX=(NLBODY-NLFACE)/2
      IF(NLMAX.LE.0) GO TO 400
      DO 300 NLINE=1,NLMAX
        WRITE(IUNIT,345) KHBUFFO
        345      FORMAT(3X,7(A3,2A6,A3))
        300      CONTINUE
C
C
C PRINT FACE LINES
C
      400 DO 400 NLINE=1,12
        DO 420 NBXD=1,7
          CALL LBOX41(KHFACE(NBXD),
-            IDIGIT(NBXD),KHRFOR(NBXD),KHRBAK(NBXD),NLINE)
        420      CONTINUE
        NLMIN=MIN0(NORDUP(NLINE),NLFACE)
        DO 440 NLREP=NLMIN,NLFACE,8
          WRITE(IUNIT,345) KHBUFFO
        440      CONTINUE
        400 CONTINUE
C
C
C IF BODY LINES BELOW FACE, RE-INITIALIZE OUTPUT BUFFER TO BACKGROUND CHARACTERS
C
      IF(NLBODY.EQ.NLFACE) GO TO 500
      DO 500 NBXD=1,7
        CALL MOVCST(KHLEFT(NBXD),(1),(21),
-          KHRBAK(NBXD),(1),(1),KHRBAK(NBXD))
      500      CONTINUE
C
C
C PRINT BODY LINES BELOW FACE
C
      NLMAX=(NLBODY-NLFACE+1)/2
      DO 600 NLINE=1,NLMAX
        WRITE(IUNIT,345) KHBUFFO
      600      CONTINUE
C
C
C DONE
C
      900 RETURN
      END

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

PROVFI  
001

```

SUBROUTINE PROVFI( 8 PRINT/OVERPRINT FILE(S)
( IUNIT, 8 UNIT NUMBER OF OUTPUT (8 OR 10 THRU 19)
-
I MROLCS, 8 LEFT MARGIN CHARACTER STRING
I LENML, 8 LENGTH (IN CHARS) OF MROLCS
I MRORCS, 8 RIGHT MARGIN CHARACTER STRING
I LENMR, 8 LENGTH (IN CHARS) OF MRORCS
I NTLCHR, 8 NEAT LINE CHARACTER
I JPSPEC, 8 PRINT SPEC (8 CHARS -- 1 PER SYMBOL POSITION)
C IF NUMERIC: NUMBER OF LINES TO ADVANCE BEFORE PRINTING
C IF NON-NUMERIC: DO NOT PRINT
I (PRTBF) 8 BUFFER TO PRINT (IN PXBDEF FORMAT)
-----
C
C
C
C HISTORY
C -----
C
C D A BECK LEC 9/25/79 REQUIREMENTS
C D A BECK LEC 10/16/79 ALGORITHM DESIGN
C D A BECK LEC 10/23/79 ALGORITHM CODING
C
C
C METHOD
C -----
C
C CHECK IF (PRTBF) IS NOT BIN TYPE INTEGER. CHECK IF THE
C NUMBER OF BINS TO BE PRINTED IS GREATER THAN OR EQUAL TO
C THE NUMBER OF OUTPUT PRINT COLUMNS. CHECK IF LENML AND
C LENMR IS LESS THAN 0. CHECK IF THE NUMBER OF OUTPUT PRINT
C COLUMNS ALLOWABLE IS GREATER THAN 132. CHECK IF LENML + 2
C (LENGTH OF 2 NTLCHR'S) IS GREATER THAN OR EQUAL TO THE
C NUMBER OF OUTPUT PRINT COLUMNS ALLOWABLE. CHECK IF THE
C MAXIMUM NUMBER OF ALTERNATE PRINT FILES WILL BE EXCEEDED.
C CHECK IF IUNIT IS NOT EQUAL TO (8 OR 10 THRU MAXIMUM
C ALLOWED).
C LOOP THRU THE INPUT SYMBOL BUFFER OBTAINING EACH SYMBOL
C PRINT BUFFER.
C CHECK IF LAST SYMBOL PRINT BUFFER AND THE MAXIMUM
C NUMBER OF COLUMNS ALLOWABLE IS GREATER THAN OR EQUAL
C TO (LENML + LENMR + 2) + THE NUMBER OF COLUMNS TO BE
C PRINTED. THEN ASSIGN LENMR TO THE LENGTH OF THE RIGHT
C MARGIN TO BE PRINTED. ELSE ASSIGN LENMR TO ZERO.
C ASSIGN POINTERS TO THE LOW AND HIGH SYMBOLS OF THE
C SYMBOL PRINT BUFFER TO BE PRINTED.
C ASSIGN UNIT NUMBER.
C CALL PROVSY TO PRINT THE SYMBOL PRINT BUFFER.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C NONE.
C
C
C EXTERNAL REFERENCES

```

PROVF 1  
002

**N-247**

ORIGINAL PAGE IS  
OF POOR QUALITY

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

PROVF1  
003

INTEGER NBINPO	% NUMBER OF BINS PER PAGE FOR DATA
INTEGER NALTPR	% NUMBER OF ALT PRINT FILES TO PRINT BUFFER
INTEGER LUNIT	% LAST VALID ALTERNATE PRINT FILE
INTEGER HROLEN	% LENGTH OF RIGHT MARGIN (ZERO FOR NO MARGIN)

```

C
C
C PROCEDURE
C -----
C
C
C CHECK FOR VALID BIN TYPE
C
  IF((IPRTBF(PXBINT).NE.'INT') CALL HDFATL(
    - ' (IPRTBF BIN TYPE NOT INT IN PROVF1)')
C
C
C CHECK FOR DATA IN BUFFER
C
  NBNDAT=(IPRTBF(PXHBIN)-IPRTBF(PXLBIN))+1 % NUMBER OF BINS IN BUFFER
  IF(NBNDAT.LE.0) CALL HDFATL(
    - ' (IPRTBF EMPTY IN PROVF1)')
C
C
C CHECK FOR INVALID MARGIN LENGTH(S)
C
  IF((LENHL.LT.0).OR.(LENHR.LT.0)) CALL HDFATL(
    - ' (NEGATIVE MARGIN LENGTH IN PROVF1)')
C
C
C CHECK FOR VALID PRINTER WIDTH
C
  IF(KPAGE.GT.132) CALL HDFATL(
    - ' (KPAGE > 132 IN PROVF1)')
C
C
C CHECK IF LEFT MARGIN AND TWO NEAT LINE CHARS. WHICH ARE REQUIRED
  EXCEED THE PRINTER WIDTH
C
  NBINPO=KPAGE-(LENHL+2) % COMPUTE THE NUMBER OF BINS PER PAGE FOR DATA
  IF(NBINPO.LE.0) CALL HDFATL(
    - ' (MARGIN TOO WIDE IN PROVF1)')
C
C
C CHECK IF THE ENTIRE BUFFER CAN BE PRINTED ON THE MAXIMUM NUMBER OF
  ALTERNATE PRINT FILES ALLOWABLE
C
  NALTPR=(NBNDAT-1)/NBINPO+1 % COMPUTE NUMBER OF ALTERNATE PRINT FILES
  IF(NALTPR.GT.NALTM) CALL HDFATL(
    - ' (IPRTBF NEEDS MORE THAN NALTM PRINT FILES IN PROVF1)')
C
C
C CHECK FOR VALID UNIT NUMBER
C
  LUNIT=0
  IF((LUNIT.EQ.0) .OR. LUNIT.GT.500)
    LUNIT=10+NALTM-1

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PROVFI  
884

```

      IF((IUNIT.LT.10).OR.(IUNIT.GT.LUNIT)) CALL HDFATL(
      '('.CDS4IN(IUNIT,(1)).' IS BAD UNIT IN PROVFI')
800 IF(NDTOTL.NE.0) GO TO 900
C
C
C SEPARATE IPRTSF INTO ALTERNATE PRINT UNITS
C
      NUNIT=IUNIT
      IPLBIN=IPRTSF(PXLBIN)
      IPMBIN=IPRTSF(PXMBIN)
      DO 800 NBINLO=IPLBIN,IPMBIN,NBINPO
      NBINHI=MIN0(NBINLO+NBINPO-1,IPMBIN)
      MROLEN=0
      IF((NBINHI-NBINLO+1)*LENMR.LE.NBINPO) MROLEN=LENMR
      CALL PROVSY(NUNIT,MROLC5,LENML,MRORC5,MROLEN,NTLCHR,JPSPEC,
      & IPRTSF(PXBINS),NBINLO,NBINHI)
      NUNIT=NUNIT+1
      IF(NUNIT.GT.LUNIT) NUNIT=10
800 CONTINUE
C
C
C RETURN TO CALLING ROUTINE
C
800 RETURN
      END

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PROVSY  
001

```

SUBROUTINE PROVSY( 8 PRINT/OVERPRINT SYMBOL BUFFER
( IUNIT,      8 OUTPUT UNIT NUMBER
-
I MROLCS,     8 LEFT MARGIN CHARACTER STRING
I LENML,      8 LENGTH (IN CHARS) OF MROLCS
I MRORCS,     8 RIGHT MARGIN CHARACTER STRING
I LENMR,      8 LENGTH (IN CHARS) OF MRORCS
I NTLCHR,     8 NEAT LINE CHARACTER
I JPSPEC,     8 PRINT SPECS (8 CHARS--1 PER SYMBOL POSITION)
C              NUMERIC CHARACTERS--
C              NUMBER OF LINES TO ADVANCE BEFORE PRINTING
C              NON-NUMERIC CHARACTERS--
C              DON'T PRINT
I IPRBUF,     8 BUFFER NOT IN PKDEF FORMAT
( NMDLO,      8 LOW WORD IN BUFFER
( NMDHI)      8 HIGH WORD IN BUFFER
-----
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      08/29/79      ALGORITHM CODING
C      J C CRISP          LEC      10/17/79      REVISE BUFFER STRUCTURE
C                                          AND INCLUDE VARIABLE MARGINS
C
C METHOD
C -----
C
C      INITIALIZE ALTERNATE PRINT FILE NAME. EXAMINE FIRST PRINT SPEC.
C      PLACE LEFT MARGIN, NEAT LINE CHARACTER, SYMBOLS AND NEAT LINE
C      CHARACTER INTO PRINT BUFFER. CHECK RIGHT MARGIN LENGTH. IF (>) 0
C      INSERT RIGHT MARGIN INTO PRINT BUFFER. IF PRINT BUFFER IS ALL
C      BLANKS. LOOP BACK FOR NEXT PRINT SPEC. ELSE CHECK UNIT NUMBER
C      AND WRITE PRINT BUFFER. LOOP BACK FOR NEXT PRINT SPEC. CONTINUE
C      UNTIL ALL PRINT SPECS HAVE BEEN APPLIED.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      ASSUMES 8 CHARACTERS PER WORD
C      UTILIZES UNIVAC EXEC 8 ER PRINTS AND ER PRINTAS
C
C EXTERNAL REFERENCES
C -----
C
C      ERPRINT      8 PRINT IMAGE ON TTY/CRT/LINE PRINTER
C      ERPRTA      8 WRITE IMAGE TO ALTERNATE PRINT FILE
C      CST4IN      8 CHARACTER STRING FOR INTEGER
C      GETICE      8 INTEGER CHARACTER EQUIVALENT FROM CHARACTER STRING
C      MOVCS      8 MOVE CHARACTER STRING
C      PUTCHR      8 PUT CHARACTER INTO CHARACTER STRING
C      MOVCHR      8 MOVE CHARACTER

```

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

PROVS7  
000

```

C      MDPATL      8 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C      INTEGER ICE      8 INTEGER CHARACTER EQUIVALENT
C      INTEGER NINVC      8 NUMBER OF INTEGERS FOR CHARACTERS
C      INTEGER LCHNE      8 LOCATE CHARACTER NOT EQUAL
C
C
C      EXCEPTIONS
C      -----
C
C      1. IF THE NUMBER OF BINS TO BE PRINTED PLUS THE LENGTH OF THE
C      LEFT AND RIGHT MARGINS PLUS THE TWO NEAT LINE CHARACTERS EXCEEDS
C      KPAGE, A QUEUED FATAL DIAGNOSTIC WILL BE ISSUED.
C
C
C      GLOBAL DECLARATIONS
C      -----
C
C      INCLUDE KONXGT.LIST      8 COMMON PROGRAM EXECUTION SWITCHES,COUNTERS
C
C
C      LOCAL DECLARATIONS
C      -----
C
C      INTEGER IPRTF (1)      8 ARGUMENT
C      INTEGER NAMFIL (2)      8 ALTERNATE PRINT FILE NAME
C      INTEGER JBFOU (23)      8 OUTPUT PRINT BUFFER PLUS BLANK PADDING
C      INTEGER ICEZER      8 ICE OF ZERO CHARACTER
C      INTEGER LOCOUT      8 OUTPUT BUFFER POINTER
C      INTEGER NICE      8 INTEGER CHARACTER EQUIVALENT
C      INTEGER NSPEC      8 PRINT SPEC POSITION
C      INTEGER NMDIN      8 BIN CONTAINING CHAR BEING PACKED
C
C
C
C      PROCEDURE
C      -----
C
C      CHECK FOR OUTPUT BUFFER OVERFLOW
C
C      IF ((NMDIN-NMDLO+1)*LENML*LENMR+2).LE.KPAGE) GO TO 100
C      CALL MDPATL ('(OUTPUT BUFFER OVERFLOW IN PROVS7)')
C      GO TO 900
C
C
C      INITIALIZE FILE NAME
C
C      100 CALL CST4IN (NAMFIL,(1),(12), IUNIT,(1))
C
C
C      INITIALIZE INTEGER CHARACTER EQUIVALENT OF ZERO
C
C      ICEZER=ICE ('0')
C
C
C      EXAMINE SPEC FOR NEXT SYMBOL POSITION
C

```



BAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PROVEY  
003

```

DO 600 NSPEC=1,8
  LOCOUT=1
  CALL GETICE (NICE, JPSPEC.(NSPEC))
  NADV=NICE-ICEZER
  IF (NADV.LT.0.OR.NADV.GT.9) GO TO 600  & NOT NUMERIC-DON'T PRINT
C
C
C MOVE LEFT MARGIN AND NEAT LINE CHARACTER TO BUFFER
C IF NADV=0, PRINT SPACES-ELSE PRINT MARGIN AND CHARACTER
C
  IF (NADV.EQ.0) CALL MOVCSY (JBFOUT.(1).(LENML),
    ' '.(11).(11).' ')
  IF (NADV.NE.0) CALL MOVCSY (JBFOUT.(1).(LENML),
    MROLCS.(11).(LENML).' ')
  LOCOUT=LOCOUT+LENML
  IF (NADV.EQ.0) CALL PUTCHR (JBFOUT.(LOCOUT), ' ')
  IF (NADV.NE.0) CALL PUTCHR (JBFOUT.(LOCOUT), NTLCHR)
  LOCOUT=LOCOUT+1
C
C
C PACK CHARACTERS INTO PRINT BUFFER
C
  DO 300 NWDIN=NWDLO,NWDHI
    CALL MOVCHR (JBFOUT.(LOCOUT), 1PRTSF(NWDIN).(NSPEC))
    LOCOUT=LOCOUT+1
  300 CONTINUE
C
C
C MOVE NEAT LINE CHAR AND RIGHT MARGIN (IF LENGTH(>0) TO BUFFER
C
  IF (NADV.EQ.0) CALL PUTCHR (JBFOUT.(LOCOUT), ' ')
  IF (NADV.NE.0) CALL PUTCHR (JBFOUT.(LOCOUT), NTLCHR)
  LOCOUT=LOCOUT+1
  IF (LENMR.EQ.0) GO TO 350
  IF (NADV.EQ.0) CALL MOVCSY (JBFOUT.(LOCOUT).(LENMR),
    ' '.(11).(LENMR).' ')
  IF (NADV.NE.0) CALL MOVCSY (JBFOUT.(LOCOUT).(LENMR),
    MRORCS.(11).(LENMR).' ')
  LOCOUT=LOCOUT+LENMR
C
C
C CHECK FOR BLANK BUFFER
C
  350 IF (LCHRNE(JBFOUT.(1).(LOCOUT).' ')).EQ.0) GO TO 600
C
C INSURE BLANK PADDING IN BUFFER
C
  CALL MOVCSY (JBFOUT.(LOCOUT).(6), ' '.(11).(11).' ')
C
C
C WRITE PRINT LINE FROM OUTPUT BUFFER
C
  IF ((UNIT.NE.6) GO TO 500  & ALTERNATE PRINT FILE
    CALL ERPRINT (NADV.NI+NC(LOCOUT-1).JBFOUT)
    GO TO 600

```

**DAN PACKAGE APPENDIX H  
UTILITY ROUTINES**

**PROVSY  
000**

```
000      CALL ERPRTA (NAMFIL.NADV.HIUNC(LOCOUT-1),JDFOUT)
C
C
C LOOP BACK FOR NEXT SPEC
C
000 CONTINUE
C
C
C NORMAL RETURN
C
000 RETURN
      END
```

**ORIGINAL PAGE IS  
OF POOR QUALITY**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PRSYML  
001

SUBROUTINE PRSYML( & PRINT SYMBOL LEGEND

I NUNIT) & OUTPUT UNIT

```

C -----
C
C HISTORY
C -----
C
C     E M SCHLOSSER      LEC      08/13/74      ORIGINAL CODE IN SYMTAB
C     E M SCHLOSSER      LEC      12/29/79      RENAME & DELETE FLO FUNCTION
C     E M SCHLOSSER      LEMSCO   05/28/80      CHANGE RANGE FORMAT FROM J2 TO J3
C
C METHOD
C -----
C
C     WRITE THE CHARACTER SYMBOL LEGEND ON NUNIT.
C
C MACHINE-DEPENDENT CODE
C -----
C
C     NONE.
C
C EXTERNAL REFERENCES
C -----
C
C     MDUNIT      & PRINT HEADING ON TOP OF NEXT PAGE
C     MOVCS      & MOVE CHARACTER STRING
C     GETCHR      & GET CHARACTER FROM CHARACTER STRING
C     INTEGER LENCST & LENGTH OF CHARACTER STRING
C
C EXCEPTIONS
C -----
C
C     NONE.
C
C GLOBAL DECLARATIONS
C -----
C
C     INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES. COUNTERS
C     INCLUDE KOMKLS.LIST      & COMMON CLASSIFICATION INFO
C     INCLUDE KOMSYN.LIST      & COMMON SYMBOL TABLE
C
C LOCAL DECLARATIONS
C -----
C
C     INTEGER KURSYN      & CURRENT SYMBOL
C     INTEGER NEXSYN      & NEXT SYMBOL
C     INTEGER KSYLEN      & LENGTH IN CHARACTERS OF CURRENT SYMBOL
C     INTEGER KURCHR      & CURRENT CHARACTER IN CURRENT SYMBOL
C     INTEGER KVALLO      & LOWEST RAD/DEN/CLA/COUNT VALUE WITH CURRENT SYMBOL

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PRSYML  
002

```

      INTEGER KVALHI      8 HIGHEST RAD/DEN/CLA/COUNT VALUE WITH CURRENT SYMBOL
C
C
C PROCEDURE
C -----
C
      CALL TRACE
C
C
C PRINT SYMBOL LEGEND HEADING AND NON-CHANGEABLE SYMBOLS
C
      IF(NUNIT.NE.8) CALL MDUNIT(4,NUNIT)
      IF(KT1PIX.NE.0) WRITE(NUNIT,115)
115 FORMAT(
      & '0SYMBOLS FOR COUNT OF PIXELS DETECTED WITHIN')
      IF(KT1PIX.EQ.0) WRITE(NUNIT,125)
125 FORMAT(
      & '0SYMBOLS FOR PIXELS DETECTED WITHIN')
      IF(KLSTYP.EQ.'RAD') WRITE(NUNIT,135) LCVLO1,LCVHI1
135 FORMAT(
      & ' SPECTRAL LIMITS AND RADIANCE RANGE ('.J3.'-'J3.')')
      IF(KLSTYP.EQ.'DEN') WRITE(NUNIT,145) LCVLO1,LCVHI1
145 FORMAT(
      & ' SPECTRAL LIMITS AND DENSITY RANGE ('.J3.'-'J3.')')
      WRITE(NUNIT,155)
155 FORMAT(
      1 '0 SYMBOL  NUMBER  MEANING'//
      2 '      :                NO DATA'//
      4 '      *                PRIMARY TICK'//
      5 '      +                SECONDARY TICK')
C
C
C PRINT USER-SPECIFIED SYMBOLS
C
      KVALLO=0
      KURSYM=' '
      NEXSYM=' '
      CALL MOVCSY(KURSYM,(1),(4),
      & KSYM(KVALLO+1),(1),(4),' ')
      DO 400 KVALHI=0,1SYMH1
      CALL MOVCSY(NEXSYM,(1),(4),
      & KSYM(KVALHI+2),(1),(4),' ')
      IF(NEXSYM.EQ.KURSYM) GO TO 400
      KNOTE1=' '
      KNOTE2=' '
      IF(KURSYM.NE.' ') GO TO 160
      KNOTE1='(BL'
      KURSYM=' A'
      KNOTE2=' NK)'
160      IF(KVALLO.EQ.KVALHI) GO TO 230
      WRITE(NUNIT,175) KNOTE1,KURSYM,KNOTE2,KVALLO,KVALHI
175      FORMAT(1X,A3,A1,A3,1X,J3.'-'J3)
      GO TO 220
200      WRITE(NUNIT,205) KNOTE1,KURSYM,KNOTE2,KVALLO
205      FORMAT(1X,A3,A1,A3,3X,J3)
220      KSYLEN=LENCST(KURSYM,4)

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

PRSYNL  
003

```
                IF(KSYLEN.LT.2) GO TO 280
                DO 280 KSYLOC=2,KSYLEN
                CALL GETCHR(KURCHR, KURSYN.(KSYLOC))
                WRITE(NUNIT,255) KURCHR
                FORMAT('+.3X.A1)
255
260                CONTINUE
280                KVALLO=KVALHI+1
                KURSYN=NEXSYN
CONTINUE
RETURN
END
```

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

PRTCHR  
001

```

SUBROUTINE PRTCHR( 8 WRITE BOX CHARACTERS (7 LINES HIGH)
I IUNIT.      8 OUTPUT UNIT NUMBER
I NCHAR.      8 NUMBER OF CHARACTERS IN STRING (MAX 19)
I JSTRNG)      8 CHARACTER STRING
-----
C
C
C (E H SCHLOSSER)
C
C
C EXTERNAL SUBROUTINES/FUNCTIONS CALLED
C -----
C
C   ERCSF
C   ERREAD
C
C
C SYMBOLIC ELEMENTS REFERENCED
C -----
C
C   DAM.BOX-CHR
C
C
C   INTEGER JBX(64),K(19),JSTRNG(1)
C   EQUIVALENCE
C   1 (K(01),K01), (K(02),K02), (K(03),K03), (K(04),K04), (K(05),K05),
C   2 (K(06),K06), (K(07),K07), (K(08),K08), (K(09),K09), (K(10),K10),
C   3 (K(11),K11), (K(12),K12), (K(13),K13), (K(14),K14), (K(15),K15),
C   4 (K(16),K16), (K(17),K17), (K(18),K18), (K(19),K19)
C   CALL TRACE
C
C
C   NBIT=0
C   NWD=1
C   DO 150 I=1,19
C       IF(1.0T.NCHAR) GO TO 130
C       K(I)=FLO(NBIT.6.JSTRNG(NWD))+1
C       NBIT=NBIT+6
C       IF(NBIT.LT.36) GO TO 150
C       NBIT=0
C       NWD=NWD+1
C       GO TO 150
C   130   K(I)='88888 '
C   150 CONTINUE
C
C   CALL ERCSF(NA0,'8ADD.E DAM.BOX-CHR . ')
C
C   200 CONTINUE
C       DO 220 NWD=1,57.8
C           CALL ERREAD(5900,JBX(NWD),1)
C   220   CONTINUE
C       WRITE(IUNIT,320)
C       1   JBX(K01),JBX(K02),JBX(K03),JBX(K04),JBX(K05),
C       2   JBX(K06),JBX(K07),JBX(K08),JBX(K09),JBX(K10),
C       3   JBX(K11),JBX(K12),JBX(K13),JBX(K14),JBX(K15),
C       4   JBX(K16),JBX(K17),JBX(K18),JBX(K19)
C   320   FORMAT(A6,18(1X,A6))

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**PRTCHR  
002**

**00 TO 200  
C  
900 RETURN  
END**

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

PRTINC  
001

```

SUBROUTINE PRTINC( 3 SPECIFY PRINT LINES/INCH & HOW CONTROLLED
I NUNIT, 3 LOGICAL UNIT NUMBER
I LPRIIN, 3 PRINT LINES PER INCH
I KONPRT) 3 TYPE OF PRINTER CONTROL:
C          'AUT' -- AUTOMATIC SOFTWARE CONTROL
C          'MAN' -- MANUAL OPERATOR CONTROL
C          -----
C (E H SCHLOSSER)
C
C      INTEGER LINCTL(5)
C
C      CALL TRACE
C
C
C      LINCTL(1) = '
      IF(KONPRT.EQ.'AUT') ENCODE(115,LINCTL) NUNIT,LPRIIN
115 FORMAT(J2.10X,'B','11.15X)
      IF(KONPRT.EQ.'MAN') ENCODE(125,LINCTL) NUNIT,LPRIIN
125 FORMAT(J2.10X,'S.CHANGE TO '.12.' LPI')
      IF(LINCTL(1).EQ.'') GO TO 900
          IF(NUNIT.EQ.6) CALL ERPRCN(3,LINCTL(3))
          IF(NUNIT.NE.6) CALL ERPRCA(5,LINCTL(1))
900 RETURN
      END

```



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PRTHRO  
001

SUBROUTINE PRTHRO( 8 SET PRINT MARGINS & LINES PER PAGE  
1 NUNIT, 8 LOGICAL UNIT NUMBER  
1 LPRIP0, 8 PRINT LINES PER PAGE (TOTAL INCLUDING LPRTOP & LPRBOT)  
1 LPRTOP, 8 PRINT LINES IN TOP MARGIN  
1 LPRBOT) 8 PRINT LINES IN BOTTOM MARGIN

C

C

C (E H SCHLOSSER)

C

INCLUDE KONXGT.LIST  
INTEGER MARGCN(4)

C

CALL TRACE  
ENCODE(115,MARGCN) NUNIT,LPRIP0,LPRTOP,LPRBOT  
115 FORMAT(J2,10X,'M.',J3,2(' ',J2),'.')  
IF(NUNIT.EQ.8) GO TO 200  
CALL ERPRCA(4,MARGCN(1))  
GO TO 900  
200 CALL ERPRCN(2,MARGCN(3))  
MLTOP8=LPRTOP  
900 RETURN  
END

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

PSTART  
001

SUBROUTINE PSTART( 8 INITIALIZE PROGRAM

1 NAMPRO) 8 PROGRAM NAME AND VERSION (CHAR STRING 24 CHARS OR LESS)

```

C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      05/12/73      ORIGINAL CODE
C      E M SCHLOSSER      LEC      02/07/74      ADD DAM LOG FILE
C      E M SCHLOSSER      LEC      01/07/79      ADD MACDAM FILE
C      E M SCHLOSSER      LEC      09/28/79      ADD DIAONOSTIC QUEUE FILE
C
C
C METHOD
C -----
C
C      INITIALIZE 8XQT MODES, LOG FILE PACKET, DIAGNOSTIC QUEUE FILE PACKET.
C      ASSION LOG FILE AND UPDATE POINTERS IN HEADER TO RESERVE 4 LOG FILE
C      RECORDS FOR CURRENTLY EXECUTING PROGRAM. IF FIRST PROGRAM OF RUN.
C      APPEND CHARACTERS FROM RIGHT OF ORIGINAL RUNID TO RIGHT OF ORIGINAL
C      QUALIFIER AND 8USE MACDAM FOR TPF8 IF NO OTHER MACDAM IS ASSIGNED.
C      INITIALIZE PROGRAM ID/DATA/TIME/HEADING 8 LOG RECORDS.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      UNFORTUNATELY.
C
C
C EXTERNAL REFERENCES
C -----
C
C      ERPCY
C      EREXIT
C      ERPRCN
C      ERION
C      ERCSF
C      ERREAD
C      ERTHAT
C      HDWARN
C      HOFATL
C      EROATE
C      ERPRNT
C      ERERR
C
C
C RESTRICTIONS
C -----
C
C 1. THE FIRST EXECUTABLE STATEMENT IN EVERY DAM PACKAGE MAIN PROGRAM MUST BE:
C      CALL PSTART('PROGRAM-NAME-AND-VERSION')
C
C 2. MAXIMUM SIZE FOR PROGRAM-NAME-AND-VERSION IS 24 CHARACTERS (PERIODS NOT

```

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

PSTART  
002

```

C   ALLOWED!!!
C
C 3. THE LAST STATEMENT EXECUTED IN EVERY DAM PACKAGE MAIN PROGRAM MUST BE:
C   CALL PSTOP --OR-- CALL PABORT
C
C
C OVERLAY RESTRICTIONS
C -----
C
C XQT & LOG LABELLED COMMON (FORPROCS) MUST BE IN THE ROOT SEGMENT.
C
C
C
C   INCLUDE KONXQT.LIST
C   INTEGER PCTSEC(28),JPCT(25)
C   EQUIVALENCE (PCTSEC(1),LOROLD), (PCTSEC(4),JPCT(1))
C   INCLUDE XQTOEF.LIST
C   INCLUDE KOMLOO.LIST
C   INCLUDE KOMIO.LIST
C   INCLUDE FACBIT.LIST
C   INCLUDE ASHDEF.LIST
C   INTEGER NAMPRO(1)
C
C
C SET MODES FROM XQT OPTIONS
C
C
C   CALL SETQWD      3 DAM PACKAGE MUST BE IN QUARTER-WORD MODE!!!
C   CALL ERPCT(25,JPCT)
C   MBATCH=XQTOPT('BATCH')
C   IF(ASHS2(JPCT(25)).NE.4) MBATCH=1
C   MCHECK=XQTOPT('CHECK')
C   MDATA=XQTOPT('DATA')*MCHECK
C   MTRACE=XQTOPT('TRACE')
C   CALL TRACE
C
C
C CHECK IF XQT OPTION E IS SET AND IF PREVIOUS PROGRAM ERROR TERMINATED
C
C   IF(XQTOPT('ERROR').EQ.0) GO TO 110      3 NO E OPTION
C   IF(ASHT3(JPCT(17)).EQ.2) GO TO 110      3 PREV PROG TERMINATED IN ERROR
C   IF(MDATA.NE.0) CALL ERCSF(NA0,'3ADD DAM.DATA-CHECK . ')
C   CALL EREXIT
C
C
C CHECK IF DAM PROGRAM FILE IS ASSIGNED
C
C   110 CALL ERCSF(NA0,'3ASO.A DAM. . ')
C   IF(ALREDY(NA0).EQ.0) GO TO 117
C
C
C CHECK IF PROGRAM WAS INITIATED AS A PROCESSOR
C
C   CALL ERCSF(NA0,'3ADD DAM.EOF . ')
C   CALL ERREAD($120,LBUF,R,ITEMP)
C   CALL ERREAD($117,LBUF,R,ITEMP)

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

PSTART  
003

```

117 CALL ERPRNT(1,3,'PROGRAM NOT FOUND')
    IF(INBATCH.NE.0) CALL ERRR
    CALL EREXIT
C
C
C DEFINE INTERNAL NAME FOR LOG FILE
C
120 IF(LOCSF(2).EQ.'USE 1.') GO TO 130
    CALL MDATL('PROGRAM ERROR -- SECOND CALL TO PSTART')
    GO TO 790
130 CALL ERCSF(NA0,LOCSF)
C
C
C INITIALIZE I/O PACKET FOR LOG FILE HEADER
C
    IOSIZE(LOOPKT)=5
    IOADDR(LOOPKT)=LOC(LBUFR)
    IOSECT(LOOPKT)=0
C
C
C ASSIGN DIAGNOSTIC QUEUE FILE & INITIALIZE I/O PACKET
C
    CALL ERCSF(NTMP,'ASO.T MDQUEUE. . ')
    IOSIZE(LUOPKT)=22
    IOADDR(LUOPKT)=LOC(LBUFR)
    IOSECT(LUOPKT)=0
C
C
C TRY TO ASSIGN OLD LOG FILE
C
    LOPHI=0
    LOCSF(2)='ASO.A'
    IF(LONSEC.LT.100) LOCSF(2)='ASO.T'      & NO PERMANENT LOG FILE
    CALL ERCSF(NASOA,LOCSF)
    IF(ROLOUT(NASOA).NE.0) CALL ROLLIN
    IF(LONSEC.LT.100) GO TO 600
    IF(REJECT(NASOA))..200
150 IF(NEHLY(NASOA))..300      & ALWAYS LOG FIRST PROGRAM OF RUN
    IF(XGTOPT('ZAPLOG'))..700  & DON'T LOG
    GO TO 300
C
C
C ASSIGN NEW LOG FILE
C
200 LOCSF(2)='ASO.CP'
    CALL ERCSF(NASOCP,LOCSF)
    IF(REJECT(NASOCP))..500      & GIVE UP!
    IF(PRCAT(NASOCP))..150      & PREVIOUSLY CATALOGED 17443
    LOMAX=4*(LONSEC/4)-4
    LOPLO=32-LOMAX
    LOPHI=4
    LOROLD=4      & FIRST PROGRAM OF FIRST RUN
    LORMI=LOPHI
    GO TO 400
C
C

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PSTART  
004

C GET EXCLUSIVE WRITE ACCESS TO LOG FILE HEADER ONLY

C

```

300 LTRY=0
310 IF(LTRY.GT.5) GO TO 500      & GIVE UP!
    LREAD=0
320 IF(LREAD.GT.10) GO TO 330    & FORCE ACCESS
    IOFUNC(LOOPKT)='BK'        & READ
    CALL ERIOH(LOOPKT)
    LREAD=LREAD+1
    IF(LRUNID.NE.'$NONE$') GO TO 320
330 LTRY=LTRY+1
    LRUNID=JPCT(2)              & GENERATED RUNID IS UNIQUE
    IOFUNC(LOOPKT)='BC'        & WRITE
    CALL ERIOH(LOOPKT)
    IOFUNC(LOOPKT)='BK'        & READ
    CALL ERIOH(LOOPKT)
    IF(LRUNID.NE.JPCT(2)) GO TO 310    & HAS ANOTHER RUN CHANGED IT?
    IOFUNC(LOOPKT)='BK'        & READ
    CALL ERIOH(LOOPKT)
    IF(LRUNID.NE.JPCT(2)) GO TO 310    & CHECK AGAIN TO BE SURE!

```

C

C

C UPDATE PROGRAM POINTERS TO RESERVE 4 LOG RECORDS

C

```

    LOPLO=MOD(LOPLO,LOMAX)+4
    LOPHI=MOD(LOPHI,LOMAX)+4

```

C

C

C UPDATE RUN POINTERS (ONLY FIRST PROGRAM OF RUN)

C

```

    IF(IALREDY(NASOA))..400
    LOROLD=LORHI
    LORHI=LOPHI

```

C

C

C WRITE UPDATED POINTERS & RELINQUISH EXCLUSIVE WRITE ACCESS

C

```

400 LRUNID='$NONE$'
    IOFUNC(LOOPKT)='BC'        & WRITE
    CALL ERIOH(LOOPKT)

```

C

C

C CATALOG NEW LOG FILE

C

```

    IF(ACCEPT(NASOA))..600      & OLD LOG FILE
    LOC$F(2)='FREE.R'          & RETAIN INTERNAL FILE NAME
    CALL ERCSF(NAO,LOC$F)
    LOC$F(2)='ASO.A'
    CALL ERCSF(NAO,LOC$F)
    IF(ACCEPT(NAO))..600      & CATALOGED PROPERLY

```

C

C

C FLAG LOG FILE CONFLICT

C

```

500 CALL NOWARN('LOG FILE CONFLICT')
    LOPHI=0      & DON'T LOG

```

**PSTART**  
**005**

**N-265**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PSTART  
000

```

C
C IF(LOPIOT.LT.4) GO TO 700
C IOSIZE(LOOPKT)=16      8 BUFFER SIZE FROM HERE ON
C IOADDR(LOOPKT)=LOC(JPIOT)
C IOSECT(LOOPKT)=LOPIOT
C IOFUNC(LOOPKT)='BC'      8 WRITE
C CALL ERION(LOOPKT)
C IOADDR(LOOPKT)=LOC(LBUFR)      8 BUFFER ADDRESS FROM HERE ON
C DO 700 N=1,16
700 LBUFR(N)=' '
C IOSECT(LOOPKT)=LOFATL
C IOFUNC(LOOPKT)='BC'      8 WRITE
C CALL ERION(LOOPKT)
C IOSECT(LOOPKT)=LOTERM
C IOFUNC(LOOPKT)='BC'      8 WRITE
C CALL ERION(LOOPKT)
C
C C WRITE PROGRAM ID/DATE/TIME ON UNIT 6
C
C 700 WRITE(6,795)
C 795 FORMAT(6X,'1 ' , ' ')      8 PAGE EJECT (8 BATCH MD0)
C IF(MBATCH.EQ.0) WRITE(6,797) JRPLOT      8 DEMAND MD0
C 797 FORMAT(10A6)
C
C C PREPARE TO READ DEFAULT COMMANDS
C
C NCARD=-9999
C
C RETURN
C
C C
C C
C C
C C
C C
C SUBROUTINE ROLLIN
C
C MINITS=6*MBATCH*20
C DO 200 N=1,MINITS
C WRITE(6,125)
C 125 FORMAT(' WAIT -- LOO FILE UNLOADED')
C CALL ERTMAT(30000)
C WRITE(6,145)
C 145 FORMAT(6X)
C CALL ERTMAT(30000)
C CALL ERCSF(1NASOA,LCSSF)
C IF(1ROLOUT(1NASOA).EQ.0) GO TO 900
C 200 CONTINUE
C LCSSF(1)='FREE.R'      8 BUT DON'T RELEASE THE 8USE.1.
C CALL ERCSF(1NASOA,LCSSF)
C LONSEC=0
C LCSSF(2)='ASO.T'
C CALL ERCSF(1NASOA,LCSSF)

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PSTART  
007

000 RETURN

C  
C  
C  
C  
C  
C  
C

SUBROUTINE DATIME

CALL CROATE(JMDY,JMNS)  
FLO(00,12,JDMN)=FLO(12,12,JMDY)  
FLO(12,12,JDMN)=FLO(00,12,JMNS)  
FLO(24,12,JDMN)=FLO(12,12,JMNS)  
ENCODE(18,100,JMDYHN) JMDY,JDMN,JMDY,JMNS,JDMN  
100 FORMAT(1X,A2,'/'',A2,'/'',R2,3X,A2,'/'',R2,1X)  
RETURN  
END



DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

PSTOP  
001

```

SUBROUTINE PSTOP( 8 TERMINATE PROGRAM
-
I MESSAGE) 8 MESSAGE (FIRST CHAR IS FORTRAN CARRIAGE CONTROL)
-----
C
C
C (E H SCHLOSSER)
C
C THIS SUBROUTINE HANDLES PROGRAM IDENTIFICATION/DATE/TIME AND LOG FILE
C ENTRIES FOR PROGRAM TERMINATION.
C
C
C
C RESTRICTIONS
C -----
C
C 1. THE FIRST EXECUTABLE STATEMENT IN EVERY DAM PACKAGE MAIN PROGRAM MUST BE:
C CALL PSTART( 'PROGRAM-NAME-AND-VERSION')
C
C 2. THE LAST STATEMENT EXECUTED IN EVERY DAM PACKAGE MAIN PROGRAM MUST BE:
C CALL PSTOP --OR-- CALL PABORT
C
C
C
C EXTERNAL SUBROUTINES/FUNCTIONS CALLED
C -----
C
C     ERSUPS
C     ERION
C     ERCSF
C     ERTRAN      8 UNIVAC SYSTEM ROUTINE
C
C
C
C OVERLAY RESTRICTIONS
C -----
C
C XQT 8 LOG LABELLED COMMON (FORPROCS) MUST BE IN THE ROOT SEGMENT.
C
C
C
C
C INCLUDE KOMXQT.LIST
C EQUIVALENCE (JMDY, LOROLD) 8 PCT SECTOR = LOROLD. . . JPCT(1), . . . JPCT(25)
C INCLUDE KOMLOG.LIST
C INCLUDE KOMIO.LIST
C DIMENSION MESSAGE(1)
C DIMENSION JTERM(3)
C DATA JTERM/' ABORT', ' ERROR', 'NORMAL'/
C CALL TRACE( 'PSTOP')
C
C
C
C
C NTERM=3-MINO(MDFATL,1)
C GO TO 800
C
C

```

C-4

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

PSTOP  
002

```

C
C
C      ENTRY PABORT  & ABORT PROGRAM
C      -----
C
C      CALL TRACE(  'PABORT')
C      NTERM=1
C      IF(MDATAC.NE.0) NTERM=2      & ERROR INSTEAD OF ABORT IN DATA/CHECKOUT MODE
C
C
C      HOUSEKEEPING
C
C      800 CALL ERTRAN(  7,NTERM)  & PUT TERMINATION STATUS IN T3 OF CONDITION WORD
C      IF(MCARD.GT.0) CALL CLOSE4(  NCARD)  & TEMP RECALL FILE USED BY READ5
C      CALL DATIME
C
C
C      WRITE TERMINATION ENTRY IN DAM PACKAGE LOG FILE
C
C      IF(LOTERM.LT.4) GO TO 820
C      ENCODE(96.835,LBUFR) JTERM(NTERM),JMDYHM,NDFATL
C      CALL ERSUPS(LBUFR(16)  )      & ACCUMULATED SUPS (200 USEC INCR)
C      IOSIZE(LOOPKT)=16
C      IOSECT(LOOPKT)=LOTERM
C      IOFUNC(LOOPKT)='8C'      & WRITE
C      CALL ERLOW(LOOPKT)
C
C
C
C      WRITE STANDARD TERMINATION MESSAGE
C
C      820 WRITE(6,825)
C      825 FORMAT('0'/'0')
C      WRITE(6,835) JTERM(NTERM),JMDYHM,NDFATL
C      835 FORMAT(1X,A6,' TERMINATION',5X,3A6,
C      1 3X,12,' FATAL ERRORS')
C      WRITE(6,855)
C      855 FORMAT('0'/'0'/'0')
C
C
C
C      WRITE SPECIAL TERMINATION MESSAGE
C
C      DO 870 NHDS=0,16
C      IF(MESSAGE(NHDS+1).EQ.-0) GO TO 890      & STOP WORD IS -0
C      870 CONTINUE
C      GO TO 900
C      890 IF(NHDS.EQ.0) GO TO 900
C      WRITE(6,895) (MESSAGE(N),N=1,NHDS)
C      895 FORMAT(16A6)
C
C
C
C      TERMINATE
C
C      900 CALL ERCSF(N,  '8FREE.A 1. . .')      & RELEASE '8USE 1. . . ' ONLY
C      IF(NTERM.EQ.2) NTERM=3      & DO NOT USE EXEC 8 ERRS TERMINATION
C      IF(MDATAC.NE.0) CALL ERCSF(N,  '8ADD DAM.DATA-CHECK . ')
C      CALL ERTRAN(  NTERM)

```

**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**PSTOP  
003**

**RETURN**

**C  
C  
C  
C  
C  
C**

**SUBROUTINE DATIME  
CALL ETRAN(9,JMOY,JHMS)  
FLO(00,12,JOHN)=FLO(12,12,JMOY)  
FLO(12,12,JOHN)=FLO(00,12,JHMS)  
FLO(24,12,JOHN)=FLO(12,12,JHMS)  
ENCODE(10,100,JMOYHM) JMOY,JOHN,JMOY,JHMS,JOHN  
100 FORMAT(1X,A2,'/'A2,'/'R2,3X,A2,'/'R2,1X)  
END**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PXBDMF  
001

```

SUBROUTINE PXBDMF( 8 DUMP PXBDEF PREAMBLE & LOW/HIGH BIN VALUES
  1 IPXBUF)      8 BUFFER IN PXBDEF FORMAT
-----
C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      11/02/79      RQMTS/DESIGN
C      J C CRISP          LEC      11/23/79      CODE/TEST
C
C
C
C METHOD
C -----
C
C      ENCODE CONTENTS OF PIXEL BUFFER PREAMBLE AND PRINT THEM OUT.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      ERPRNT      8 PRINT ON TTY OR LINE PRINTER
C      CBINIT      8 INITIALIZE CHARACTER BUFFER
C      CB4IN       8 CHARACTER BUFFER FOR INTEGER
C      CB4CST      8 CHARACTER BUFFER FOR CHARACTER STRING
C      INTEGER NIN  8 NUMBER OF INTEGERS FOR NUMBER OF CHARACTERS
C      INTEGER LENCST 8 LENGTH OF CHARACTER STRING OR CHARACTER BUFFER
C
C
C EXCEPTIONS
C -----
C
C      1. RESULTS ARE UNDEFINED IF IPXBUF IS NOT IN PXBDEF FORMAT.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE PXBDEF.LIST      8 DEFINE STRUCTURE OF PIXEL BUFFER
C      INCLUDE ICBUF1.LIST     8 DECLARE CHARACTER BUFFER
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER IPXBUF(1)      8 ARGUMENT
C      INTEGER IPXLO          8 VALUE OF LOW BIN
C      INTEGER IPXHI          8 VALUE OF HIGH BIN
C

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PXBONP  
002

```

C
C PROCEDURE
C -----
C
C     CALL TRACE
C
C
C PRINT NAMES OF FIELDS IN BUFFER PREAMBLE
C
C     CALL ERPRNT(1,N14NC(75),
C     * RECN LINO CHAN QUAL BINT LBIN LS/C MBIN MS/C NOIN NODA LJOI HJOI
C     & LVAL MVAL')
C
C
C ENCODE CONTENTS OF BUFFER PREAMBLE
C
C     CALL CBINIT(ICBUF1)
C     CALL CB4IN (ICBUF1, IPXBUF(PXRECN).(5))
C     CALL CB4IN (ICBUF1, IPXBUF(PXLINO).(5))
C     CALL CB4IN (ICBUF1, IPXBUF(PXCHAN).(5))
C     CALL CB4IN (ICBUF1, IPXBUF(PXQUAL).(5))
C     CALL CB4CST(ICBUF1, ' ',(1),(1))
C     CALL CB4CST(ICBUF1, IPXBUF(PXBINT).(1),(4))
C     CALL CB4IN (ICBUF1, IPXBUF(PXLBIN).(5))
C     CALL CB4IN (ICBUF1, IPXBUF(PXLSAH).(5))
C     CALL CB4IN (ICBUF1, IPXBUF(PXHBIN).(5))
C     CALL CB4IN (ICBUF1, IPXBUF(PXHSAH).(5))
C     CALL CB4IN (ICBUF1, IPXBUF(PXNOIN).(5))
C     CALL CB4IN (ICBUF1, IPXBUF(PXNODA).(5))
C     CALL CB4IN (ICBUF1, IPXBUF(PXLJOI).(5))
C     CALL CB4IN (ICBUF1, IPXBUF(PXHJOI).(5))
C
C
C ENCODE CONTENTS OF LOW & HIGH BINS
C
C     IF(IPXBUF(PXBINT).NE.'CHR') GO TO 220
C         CALL GETICE(IPXLO, IPXBUF(PXBINS).(IPXBUF(PXLBIN)))
C         CALL GETICE(IPXHI, IPXBUF(PXBINS).(IPXBUF(PXHBIN)))
C         GO TO 290
C     220 IF(IPXBUF(PXBINT).NE.'BYT') GO TO 240
C         CALL GETBYT(IPXLO, IPXBUF(PXBINS).(IPXBUF(PXLBIN)))
C         CALL GETBYT(IPXHI, IPXBUF(PXBINS).(IPXBUF(PXHBIN)))
C         GO TO 290
C     240 IF(IPXBUF(PXBINT).NE.'INT') GO TO 260
C         IPXLO=IPXBUF(PXBINS-1+PXLBIN)
C         IPXHI=IPXBUF(PXBINS-1+PXHBIN)
C         GO TO 290
C     260 GO TO 300
C     280 CALL CB4CST(ICBUF1, ' ',(1),(1))
C         IF(IABS(IPXLO).GT.9999) CALL CB4CST(ICBUF1, IPXLO.(1),(4))
C         IF(IABS(IPXLO).LE.9999) CALL CB4IN (ICBUF1, IPXLO.(4))
C         CALL CB4CST(ICBUF1, ' ',(1),(1))
C         IF(IABS(IPXHI).GT.9999) CALL CB4CST(ICBUF1, IPXHI.(1),(4))
C         IF(IABS(IPXHI).LE.9999) CALL CB4IN (ICBUF1, IPXHI.(4))

```

**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**PXSDMP  
003**

```
C PRINT ENCODED CONTENTS OF BUFFER PREAMBLE
C
300 CALL IPRINT(1,N14NC(LENCST(1CBUF1,60)),1CBUF1)
C
C
C DONE
C
      RETURN
      END
```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

PX4AM  
801

```

SUBROUTINE PX4AM( 8 PX8DEF PREAMBLE FOR HOP AM (UNCORRECTED NSS) BUFFER
0 MPXPRE. 8 PX8DEF FORMAT PREAMBLE
0 ITSTAT. 8 I/O STATUS CODES
C      ' NORMAL COMPLETION
C      'EOF' DATA OUTSIDE FILE
C
C
C      I HANBUF. 8 NSS PIXEL BUFFER IN HOP AM FORMAT
C      I NIIBF. 8 NUMBER OF INTEGERS IN BUFFER
C      I NCHAN. 8 CHANNEL NUMBER
C      I NSASLO. 8 LOW ADJUSTED SAMPLE NUMBER
C      I NSASHI. 8 HIGH ADJUSTED SAMPLE NUMBER
C      -----
C
C
C
C HISTORY
C -----
C
C      MARY TOMPKINS      LEC      07/27/79      REQUIREMENTS
C      J C CRISP          LEC      09/12/79      ALGORITHM DESIGN
C      J C CRISP          LEC      09/13/79      ALGORITHM CODING
C
C
C METHOD
C -----
C
C      SET PX8DEF PREAMBLE FIELDS AS FOLLOWS:
C      (PX8INT) = 'BYT'. (PX8NOD) = 129. (PX8NOIN) = 129. (PXLJ01) = 0
C      AND (PX8HJ01) = 0. BASED ON CHANNEL NUMBER, BUFFER LENGTH, AND
C      NUMBER OF DEFINED PIXELS ASCERTAIN THE RANGE OF SAMPLES IN THE
C      HANBUF BUFFER. COMPARE THE RANGE OF SAMPLES REQUESTED WITH
C      THE RANGE OF SAMPLES AVAILABLE AND SET: (PXLBIN), (PXLSAM),
C      (PXHSAM). IF THE NUMBER OF BYTES IN BUFFER >= 3567, THEN
C      NPXIL = F(BYTES 3569 & 3568) ELSE NPXIL = MIN(3000, NBY18-(13*74)).
C
C      BIN WITH      1ST      BIN WITH      LAST      BIN WITH      BIN NUMBER
C      SAMPLE 1      DEFINED 1ST      DEFINED 1ST      AS FUNCTION
C      IN      SAMPLE DEFINED      SAMPLE DEFINED      OF SAMPLE
C      AM      'AM      SAMPLE      IN      SAMPLE      NUMBER
C      BUFFER      BUFFER      AM      AM      AM
C
C      CHAN 1  13*69      7      13*75      NPXIL*8  13*75+NPXIL-1  =9*13*75-7
C      CHAN 2  13*69      9      13*73      NPXIL*4  13*73+NPXIL-1  =9*13*73-9
C      CHAN 3  13*69      3      13*71      NPXIL*2  13*71+NPXIL-1  =9*13*71-3
C      CHAN 4  13*69      1      13*69      NPXIL*0  13*69+NPXIL-1  =9*13*69-1
C      CHAN 5  13*69      1      13*69      NPXIL*0  13*00+NPXIL-1  =9*13*69-1
C
C      * IGNORING LOW RESOLUTION CHANNEL 5
C      IF NUMBER OF BYTES >= 2567, THEN (PXQUAL) = F(BYTES 3567)
C      ELSE (PXQUAL) = 4
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE
C
C EXTERNAL REFERENCES

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**PKMAN  
002**

```

C -----
C
C      GETBYT      & GET NON-NEG INTEGER FROM BYTE IN BYTE STRING
C      GETQBY      & GEN NON-NEG INTEGER FROM QUADRUPLE BYTE IN BYTE STRING
C      INTEGER NB4NI      & NUMBER OF BYTES FOR NUMBER OF INTEGERS
C
C
C EXCEPTIONS
C -----
C
C      1. IF THE REQUESTED SAMPLES ARE INSIDE THE FILE BUT OUTSIDE THE
C          BUFFER SET PXQUAL = 0. PXLSAM, PKNSAM, PXLBIN, PKMBIN, AND
C          PKHBIN EQUAL TO ZERO. SET ISTAT = 'EOF'.
C      2. IF TRUNCATION OF REQUESTED SAMPLES OCCUR PKMSAM IS ADJUSTED
C          TO REFLECT THE HIGHEST AVAILABLE PIXEL.
C      3. IF MPXPRE(PXQUAL) > 3 OR < 0. THEN MPXPRE(PXQUAL) = 4
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
C      INCLUDE PKBDEF.LIST      & DEFINITION OF INTERNAL BUFFER STRUCTURE
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER MAMBUF(NIIBF)      & ARGUMENT
C      INTEGER MPXPRE(1)          & ARGUMENT
C      INTEGER NBYIBF              & NUMBER OF BYTES IN 1 BUFFER
C      INTEGER NPXIL               & NUMBER OF PIXELS IN ONE LINE
C      INTEGER IBYT                & VALUE OF BYTE IN BYTE STRING
C      INTEGER IQUAL0/0000/        & QUALITY - GOOD DATA
C      INTEGER IQUAL1/0077/        & QUALITY - DATA BASED ON SUBSTITUTED LINE
C      INTEGER IQUAL2/0007/        & QUALITY - FILLED LINE ON INPUT
C      INTEGER IQUAL3/0070/        & QUALITY - FILLED LINE ON OUTPUT
C      INTEGER NBOFST(5)/75.73.71.69.69/ & NUMBER OF BYTES OFFSET
C      INTEGER NIDSAM(5)/7.5.3.1.1/ & NUMBER OF FIRST DEFINED SAMPLE
C      INTEGER LBOFST(5)/6.4.2.0.-69/ & LAST SAMPLE OFFSET
C
C
C PROCEDURE
C -----
C
C
C SET STATUS AND POINTERS
C
C      ISTAT = ' '
C      MPXPRE(PXQUAL) = 4
C      MPXPRE(PXBINT) = 'BYT'
C      MPXPRE(PXNODI) = 120
C      MPXPRE(PXNOIN) = 120
C      MPXPRE(PXLJOI) = 0
C      MPXPRE(PXHJOI) = 0

```



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PXVAM  
003

```

C
C SET RECORD NUMBER
C
    CALL GETQBY(NPXPRE(PXREC),  HANBU(PXBINS),1)
C
C COMPUTE NUMBER OF BYTES IN BUFFER -- IF AVAILABLE.
C GET QUALITY CODE AND # OF PIXELS PER LINE
C
    NBY10F = N84N1(N10F)
    IF(NBY10F.LT.3567) GO TO 100
C
    CALL GETBYT(IQUAL,  HANBU(PXBINS).3567)
    IF(IQUAL.EQ.IQUAL0) MPXPRE(PXQUAL) = 0
    IF(IQUAL.EQ.IQUAL1) MPXPRE(PXQUAL) = 1
    IF(IQUAL.EQ.IQUAL2) MPXPRE(PXQUAL) = 2
    IF(IQUAL.EQ.IQUAL3) MPXPRE(PXQUAL) = 3
C
    CALL GETBYT(NPXIL,  HANBU(PXBINS).3565)
    CALL GETBYT(10YT,  HANBU(PXBINS).3566)
    NPXIL = (NPXIL*2**6) + 10YT
    GO TO 150
C
C ELSE. SET NPXIL
C
    100 NPXIL = MIN0(3000,(NBY10F - (13*74)))
C
C SET LOW AND HIGH SAMPLES AND BINS
C
    150 MPXPRE(PXLSAM) = MAX0(MSASLO,NIDSAM(NCHAN))
    MPXPRE(PXLBIN) = MPXPRE(PXLSAM)+(13*NBOFST(NCHAN))-NIDSAM(NCHAN)
    MPXPRE(PXHSAH) = MIN0(MSASHI,(NPXIL+LSOFST(MSCHAN)))
    MPXPRE(PXHBIN) = MPXPRE(PXHSAH)+(13*NBOFST(NCHAN))-NIDSAM(NCHAN)
C
C CHECK FOR REQUESTED SAMPLES OUTSIDE OF BUFFER
C
    250 IF(MPXPRE(PXLBIN).LE.NBY10F) GO TO 300
    MPXPRE(PXBINT) = 'NUL'
    MPXPRE(PXQUAL) = 9
    MPXPRE(PXLSAM) = 0
    MPXPRE(PXHSAH) = 0
    MPXPRE(PXLBIN) = 0
    MPXPRE(PXHBIN) = 0
    ISTAT = 'EOF'
C
    300 RETURN
C
    END

```

DAH PACKAGE APPENDIX M  
UTILITY ROUTINES

PXVAR  
001

```

SUBROUTINE PXVAR( 8 PXDEF PREAMBLE FOR HOP AR (UNCORRECTED RBV) BUFFER
0 HXPREF. 8 PXDEF FORMAT PREAMBLE
0 ITSTAT. 8 I/O STATUS CODES
C      . . . NORMAL COMPLETION
C      'EOF' DATA OUTSIDE FILE
C
C
C      I HANBUF. 8 HSB PIXEL BUFFER IN HOP AR FORMAT
C      I NIIBF. 8 NUMBER OF INTEGERS IN BUFFER
C      I NCHAN. 8 CHANNEL NUMBER
C      I HSASLO. 8 LOW ADJUSTED SAMPLE NUMBER
C      I HSASHI) 8 HIGH ADJUSTED SAMPLE NUMBER
C      -----
C
C
C HISTORY
C -----
C
C      MARY TOMPKINS      LEC      07/27/79      REQUIREMENTS
C                        LEC      /  /      ALGORITHM DESIGN
C                        LEC      /  /      ALGORITHM CODING
C
C
C METHOD
C -----
C
C      ITSTAT = . . .
C      SET PXDEF PREAMBLE FIELDS AS FOLLOWS:
C      (PXINT) = 'BYT', (PXNOD) = 129, (PXNOIN) = 129, (PXLJOI) = 0
C      AND (PXHJOI) = 0. BASED ON BUFFER LENGTH, CHANNEL NUMBER, AND
C      NUMBER OF DEFINED PIXELS ASCERTAIN THE RANGE OF SAMPLES IN THE
C      BUFFER. COMPARE THE RANGE OF SAMPLES REQUESTED WITH THE RANGE
C      OF SAMPLES AVAILABLE AND SET: (PXLBIN), (PXMBIN), (PXLSAM), AND
C      (PXNSAM). IF NUMBER OF BYTES >= 5200, THEN (PXQUAL) = F(BYTE 5300)
C      ELSE (PXQUAL) = 4.
C
C
C EXTERNAL REFERENCES
C -----
C
C      HOPATL      8 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C
C
C EXCEPTIONS
C -----
C
C      1. IF THE REQUESTED SAMPLES ARE INSIDE THE FILE BUT OUTSIDE THE
C      BUFFER SET PXQUAL = 9. PXLSAM, PXNSAM, PXLBIN, PXMBIN, AND
C      PXHBIN EQUAL TO ZERO. SET ITSTAT = 'EOF'.
C      2. IF TRUNCATION OF REQUESTED SAMPLES OCCUR PXNSAM IS ADJUSTED
C      TO REFLECT THE HIGHEST AVAILABLE PIXEL.
C      3. IF HXPREF(PXQUAL) > 3 OR < 0, THEN HXPREF(PXQUAL) = 4
C
C
C GLOBAL DECLARATIONS
C -----

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PX4AR  
000

```
C
C      INCLUDE KONXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
C      INCLUDE PX0DEF.LIST      & DEFINITION OF INTERNAL BUFFER STRUCTURE
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER HAMBUF(NH1BF)      & ARGUMENT
C
C
C PROCEDURE
C -----
C
C      CALL TRACE
C
C      CALL HOFATL( 'PX4AR NOT YET IMPLEMENTED')
C
C 000 RETURN
C
C      END
```

PX4PH  
001

C                    •        NORMAL COMPLETION  
C                    'EOF' DATA OUTSIDE FILE

**F**

**C HISTORY**

C				
C	CHARLES HELMKE	LEC	07/27/79	REQUIREMENTS
C	J C CRISP	LEC	09/09/79	ALGORITHM DESIGN
C	J C CRISP	LEC	09/07/79	ALGORITHM CODING

```
C
C
C METHOD
C
```

```

C
C      ISTAT = ' '.
C      SET PXDEF PREAMBLE FIELDS AS FOLLOWS:
C      (PXBINT) = 'BYT', (PXNODA) = 128, (PXNOIN) = 128, (PXHJOI) = 0,
C      AND (PXRECN) = MDP RECORD NUMBER. COMPUTE ADJUSTED RFC. IF NUMBER
C      OF BYTES IN PM BUFFER >= 3580, THEN RFC = RFC, ELSE
C      RFC = MAX0(RFC, 3580 - N11BF). SET (PXLBIN), (PXHLIN), (PXLBAH),
C      (PXNSAH) OF PXDEF PREAMBLE FROM LEFT FILL COUNT AND RIGHT FILL
C      COUNT OF MDP PM PREAMBLE AS FOLLOWS:

```

	IN PH BUFFER	REQUESTED AND IN PH BUFFER
PH BUFFER BIN ASSIGNED TO SAMPLE 1	12+1	12+1
1ST DEFINED SAMPLE	(LFC+1)	MAX0(LFC+1,MSASL0)
PH BUFFER BIN CONTAINING 1ST DEFINED SAMPLE	12+(LFC+1)	12+MAX0(LFC+1,MSASH1)
LAST DEFINED SAMPLE	(3540-R0X)	MING(3540-RFC,MSASH1)
PH BUFFER BIN CONTAINING LAST DEFINED SAMPLE	(3540-RFC)+12	MING(3540-RFC,MSASH1)+12
PH BUFFER BIN ASSIGNED TO SAMPLE NUMBER 8	8+12	8+12

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**PXDPN  
002**

**C MACHINE-DEPENDENT CODE**

**C -----**

**C NONE**

**C**

**C EXTERNAL REFERENCES**

**C -----**

**C GETBYT        : GET NON-NEG INTEGER FROM BYTE IN BYTE STRING**  
**C GETNBY        : GET NON-NEG INTEGER FROM NYBLE**  
**C GETQBY        : GET NON-NEG INTEGER FROM QUADRUPLE BYTE IN BYTE STRING**  
**C        INTEGER NB\*NI        : NUMBER OF BYTES FOR NUMBER OF INTEGERS**

**C**

**C EXCEPTIONS**

**C -----**

- C 1. IF THE REQUESTED SAMPLES ARE INSIDE THE FILE BUT OUTSIDE THE**  
**C     BUFFER SET PXQUAL = 9, PXLSAM, PXMSAM, PXLBIN, PXMBIN, AND**  
**C     PXMBIN EQUAL TO ZERO. SET ISTAT = 'EOF'.**
- C 2. IF TRUNCATION OF REQUESTED SAMPLES OCCUR PXMSAM IS ADJUSTED**  
**C     TO REFLECT THE HIGHEST AVAILABLE PIXEL.**

**C**

**C GLOBAL DECLARATIONS**

**C -----**

**C INCLUDE KOMXOT.LIST        : COMMON PROGRAM EXECUTION SWITCHES, COUNTERS**  
**C INCLUDE PXBDEF.LIST        : DEFINITION OF INTERNAL BUFFER STRUCTURE**

**C**

**C LOCAL DECLARATIONS**

**C -----**

<b>C        INTEGER MPMBUF(NI*IBF)</b>	<b>: ARGUMENTS</b>
<b>C        INTEGER MPXPRE(1)</b>	<b>: ARGUMENT</b>
<b>C        INTEGER IQUAL</b>	<b>: QUALITY CODE</b>
<b>C        INTEGER NBYIBF</b>	<b>: NUMBER OF BYTES IN BUFFER</b>
<b>C        INTEGER MPHLC</b>	<b>: LEFT FILL COUNT</b>
<b>C        INTEGER MPHRC</b>	<b>: RIGHT FILL COUNT</b>
<b>C        INTEGER IQUAL0/0300/</b>	<b>: QUALITY - GOOD DATA</b>
<b>C        INTEGER IQUAL1/0011/</b>	<b>: QUALITY - DATA BASED ON SUBSTITUTED LINE</b>
<b>C        INTEGER IQUAL2/0022/</b>	<b>: QUALITY - FILLED LINE ON INPUT</b>
<b>C        INTEGER IQUAL3/0333/</b>	<b>: QUALITY - FILLED LINE ON OUTPUT</b>

**C**

**C PROCEDURE**

**C -----**

**C**

**C**

**C SET STATUS AND POINTERS**

**C**

**C        ISTAT = . . .**  
**C        MPXPRE(PXQUAL) = 9**  
**C        MPXPRE(PXBINT) = 'BYT'**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PX4PH  
003

```

      MPXPRE(PXNODI) = 129
      MPXPRE(PXNOIN) = 128
      MPXPRE(PXLJOI) = 0
      MPXPRE(PXHJOI) = 0
C
C
C SET RECORD NUMBER
C
      CALL GETQBY(MPXPRE(PXRECNI),  MPMBUF(PXBINS),1)
C
C
C SET QUALITY CODE
C
      CALL GETBYT(IQUAL,  MPMBUF(PXBINS),9)
      IF(IQUAL.EQ.IQUAL0) MPXPRE(PXQUAL) = 0
      IF(IQUAL.EQ.IQUAL1) MPXPRE(PXQUAL) = 1
      IF(IQUAL.EQ.IQUAL2) MPXPRE(PXQUAL) = 2
      IF(IQUAL.EQ.IQUAL3) MPXPRE(PXQUAL) = 3
C
C
C SET LEFT FILL COUNT
C
      CALL GETBYT(MPHLFC,  MPMBUF(PXBINS),10)
      CALL GETNYB(NYBLE,  MPMBUF(PXBINS),11,1)
      MPHLC = (MPHLC*2**4) + NYBLE
C
C
C SET RIGHT FILL COUNT AND ADJUST IF NECESSARY
C
      CALL GETNYB(NYBLE,  MPMBUF(PXBINS),11,2)
      CALL GETBYT(IBYT,  MPMBUF(PXBINS),12)
      MPHRC = (NYBLE*2**8) + IBYT
      NBYIBF = NB4NI(NIIBF)
      IF(NBYIBF.LT.3560) MPHRC = MAX0(MPHRC,3560-NBYIBF)
C
C
C SET POINTERS TO LOW AND HIGH SAMPLES AND BINS
C
      MPXPRE(PXLSAM) = MAX0(MPHLC+1,MSASLO)
      MPXPRE(PXLBIN) = MPXPRE(PXLSAM) + 12
      MPXPRE(PXHSAH) = MIN0(3540-MPHRC,MSASHI)
      MPXPRE(PXHBIN) = MPXPRE(PXHSAH) + 12
C
C
C CHECK FOR REQUESTED SAMPLES OUTSIDE OF BUFFER
C
      250 IF(MPXPRE(PXLBIN).LE.NBYIBF) GO TO 900
      MPXPRE(PXBINT) = 'NUL'
      MPXPRE(PXQUAL) = 9
      MPXPRE(PXLSAM) = 0
      MPXPRE(PXHSAH) = 0
      MPXPRE(PXLBIN) = 0
      MPXPRE(PXHBIN) = 0
      ITSTAT = 'EOF'
C
      900 RETURN

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**PXPH  
004**

**C  
END**

PX4PR  
001

**H-203**



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

PX4PR  
002

```

C
C
C MACHINE-DEPENDENT CODE
C -----
C
C     NONE
C
C
C EXTERNAL REFERENCES
C -----
C
C     GETBYT      % GET NON-NEG INTEGER FROM BYTE IN BYTE STRING
C     GETDBY      % GET NON-NEG INTEGER FROM DOUBLE BYTE
C     GETQBY      % GEN NON-NEG INTEGER FROM QUADRUPLE BYTE IN BYTE STRING
C     INTEGER NB4NI % NUMBER OF BYTES FOR NUMBER OF INTEGERS
C
C
C EXCEPTIONS
C -----
C
C     1. IF THE REQUESTED SAMPLES ARE INSIDE THE FILE BUT OUTSIDE THE
C         BUFFER SET PXQUAL = 9, PXLSAM, PXHSAM, PXLBIN, PXHBIN, AND
C         PXHBIN EQUAL TO ZERO, SET ITSTAT = 'EOF'.
C     2. IF TRUNCATION OF REQUESTED SAMPLES OCCUR PXHSAM IS ADJUSTED
C         TO REFLECT THE HIGHEST AVAILABLE PIXEL.
C     3. IF MPXPRE(PXQUAL) > 3 OR < 0, THEN MPXPRE(PXQUAL) = 4
C
C
C GLOBAL DECLARATIONS
C -----
C
C     INCLUDE KOMXOT.LIST      % COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
C     INCLUDE PXBDEF.LIST      % DEFINITION OF INTERNAL BUFFER STRUCTURE
C
C
C LOCAL DECLARATIONS
C -----
C
C     INTEGER MPRBUF(111BF)    % ARGUMENTS
C     INTEGER MPXPRE(1)        % ARGUMENT
C     INTEGER IQUAL            % QUALITY CODE
C     INTEGER NBY1BF           % NUMBER OF BYTES IN BUFFER
C     INTEGER MPRLFC           % LEFT FILL COUNT
C     INTEGER MPRRFC           % RIGHT FILL COUNT
C     INTEGER IQUAL0/0000/     % QUALITY - GOOD DATA
C     INTEGER IQUAL1/0077/     % QUALITY - DATA BASED ON SUBSTITUTED LINE
C     INTEGER IQUAL2/0007/     % QUALITY - FILLED LINE ON INPUT
C     INTEGER IQUAL3/0070/     % QUALITY - FILLED LINE ON OUTPUT
C
C
C PROCEDURE
C -----
C
C     CALL TRACE
C
C

```

**DAM PACKAGE APPENDIX M  
UTILITY ROUTINES**

**PX4PR  
003**

**C SET STATUS AND POINTERS**

```
C
  ITSTAT = ' '
  MPXPRE(PXQUAL) = 4
  MPXPRE(PXBINT) = 'JYT'
  MPXPRE(PXNOD) = 129
  MPXPRE(PXNOIN) = 129
  MPXPRE(PXLJOI) = 0
  MPXPRE(PXHJOI) = 0
```

**C  
C SET RECORD NUMBER**

```
C
  CALL GETQBY(MPXPRE(PXREC), MPRBUF(PXBINS),1)
```

**C  
C SET LEFT FILL COUNT**

```
C
  CALL GETQBY(MPRLFC, MPRBUF(PXBINS),9)
```

**C  
C SET RIGHT FILL COUNT AND ADJUST IF NECESSARY**

```
C
  CALL GETQBY(MPRRFC, MPRBUF(PXBINS),11)
  NBYIBF = NBYNI(NIIBF)
  IF(NBYIBF.LT.5337) MPRRFC = MAX0(MPRRFC,5334-NBYIBF)
```

**C  
C SET QUALITY CODE, IF AVAILABLE**

```
C
  IF(NBYIBF.LT.5337) GO TO 100
  CALL GETBYT(IQUAL, MPRBUF(PXBINS),5337)
  IF(IQUAL.EQ.IQUAL0) MPXPRE(PXQUAL) = 0
  IF(IQUAL.EQ.IQUAL1) MPXPRE(PXQUAL) = 1
  IF(IQUAL.EQ.IQUAL2) MPXPRE(PXQUAL) = 2
  IF(IQUAL.EQ.IQUAL3) MPXPRE(PXQUAL) = 3
```

**C  
C SET POINTERS TO LOW AND HIGH SAMPLES AND BINS**

```
C
  100 MPXPRE(PXLSAM) = MAX0(MPRLFC+1,MSASLO)
  MPXPRE(PXLBIN) = MPXPRE(PXLSAM) + 12
  MPXPRE(PXHSAH) = MIN0(5334-MPRRFC,MSASHI)
  MPXPRE(PXHBIN) = MPXPRE(PXHSAH) + 12
```

**C  
C CHECK FOR REQUESTED SAMPLES OUTSIDE OF BUFFER**

```
C
  250 IF(MPXPRE(PXLBIN).LE.NBYIBF) GO TO 900
  MPXPRE(PXBINT) = 'NUL'
  MPXPRE(PXQUAL) = 9
  MPXPRE(PXLSAM) = 0
  MPXPRE(PXHSAH) = 0
  MPXPRE(PXLBIN) = 0
  MPXPRE(PXHBIN) = 0
  ITSTAT = 'EOF'
```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**PRGPR  
004**

**C  
000 RETURN  
C  
END**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

QUAD  
001

```

SUBROUTINE QUAD( 8 FIT  $Y = A \cdot X^2 + B \cdot X + C$  & SOLVE FOR EXTREMUM
1 XVAL.          8 ARRAY CONTAINS 3 VALUES OF X
1 YVAL.          8 ARRAY CONTAINS 3 VALUES OF Y
0 A.             8 A COEFFICIENT
0 B.             8 B COEFFICIENT
0 C.             8 C COEFFICIENT
0 XTREME)        8 X VALUE AT EXTREMUM
-----
C
C
C (M L BROWN)
C
C
C DIMENSION XVAL(1),YVAL(1),XTREME(1)
C DOUBLE PRECISION X(3),Y(3),DEL,AA,BB,CC
C CALL TRACE
C
C DO 100 I=1,3
C   X(I)=XVAL(I)
C 100 Y(I)=YVAL(I)
C
C   DEL=X(1)**2*(X(2)-X(3))-X(1)*
C   1(X(2)**2-X(3)**2)+(X(2)**2*X(3))
C   2 -X(3)**2*X(2)
C   IF(DEL.NE.0) GO TO 200
C   CALL MOWARN('DETERMINANT = 0 IN QUAD')
C   GO TO 900
C
C
C
C CALCULATE FIRST COEFF OF QUADRATIC
C
C 200 AA=Y(1)*(X(2)-X(3))-X(1)*(Y(2)
C   1-Y(3))+(Y(2)*X(3)-Y(3)*X(2))
C   IF(AA.NE.0) GO TO 300
C   CALL MOWARN('EXTREMUM UNDEFINED IN QUAD')
C   GO TO 900
C
C
C
C CALCULATE SECOND COEFF OF QUADRATIC
C
C 300 BB=X(1)**2*(Y(2)-Y(3))-Y(1)*
C   1(X(2)**2-X(3)**2)+X(2)**2*Y(3)
C   2-X(3)**2*Y(2)
C
C
C
C CALCULATE THIRD COEFF OF QUADRATIC
C
C   CC=X(1)**2*(X(2)*Y(3)-X(3)*Y(2))-X(1)*(X(2)**2*Y(3)
C   1 -X(3)**2*Y(2))+Y(1)*(X(2)**2*X(3)-X(3)**2*X(2))
C   A=AA/DEL
C   B=BB/DEL
C   C=CC/DEL
C
C
C
C DETERMINE X VALUE AT EXTREMUM
C
C   XTREME=-BB/(2.*AA)

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**QUAD  
002**

**C  
C  
900 RETURN  
END**

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**QUARTN  
001**

```

      REAL FUNCTION
      0      8 NORMALIZED QUARTINAX ROTATION CRITERION
      = QUARTN(
      1 AMAT,      8 SINGLE PRECISION MATRIX
      ( NRUSE,NCUSE, 8 NUMBER OF ROWS & COLUMNS USED
      ( NRDIM,NCDIM, 8 NUMBER OF ROWS & COLUMNS DIMENSIONED
      ( NC1PRO,NC2PRO) 8 COLUMN NUMBERS OF COLUMNS TO PROCESS
      -----
C
C
C (E H SCHLOSSER)
C
C
      DIMENSION AMAT(NRDIM,NCDIM)
      CALL TRACE
C
C
      QUARTN=0.
      DO 240 NR=1,NRUSE
      QUARTN=QUARTN+(AMAT(NR,NC1PRO)**4+AMAT(NR,NC2PRO)**4)/
      8      (AMAT(NR,NC1PRO)**2+AMAT(NR,NC2PRO)**2)**2
240 CONTINUE
      RETURN
      END

```

**QAM PACKAGE APPENDIX H  
UTILITY ROUTINES**

**QUARTU  
001**

```

      REAL FUNCTION
      0      3 UN-NORMALIZED QUARTMAX ROTATION CRITERION
      = QUARTU(
      1 AMAT.      3 SINGLE PRECISION MATRIX
      1 NRUSE,NCUSE. 3 NUMBER OF ROWS & COLUMNS USED
      1 NRDIM,NCDIM. 3 NUMBER OF ROWS & COLUMNS DIMENSIONED
      1 NC1PRO,NC2PRO) 3 COLUMN NUMBERS OF COLUMNS TO PROCESS
      -----
C
C
C (E H SCHLOSSER)
C
C
C      DIMENSION AMAT(NRDIM,NCDIM)
C      CALL TRACE
C
C
C      QUARTU=0.
C      DO 240 NR=1,NRUSE
C      QUARTU=QUARTU+AMAT(NR,NC1PRO)**4+AMAT(NR,NC2PRO)**4
240  CONTINUE
      RETURN
      END

```

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

RD3BIL  
001

```

SUBROUTINE RD3BIL( 3 READ/CONVERT BIL FORMAT FROM TAPE TO PXBDEF FORMAT
0 MPXBUF. 3 NSS PIXEL BUFFER IN PXBDEF FORMAT
1 NHIBF. 3 NUMBER OF WORDS IN 1 BUFFER
0 ISTAT. 3 I/O STATUS CODE      '      ' NORMAL COMPLETION
                                'EOF' ' END OF FILE OR REQUESTED DATA
                                '      ' OUT OF FILE
                                'BADR' BAD RECORD
                                'BADF' BAD FILE
                                'OFL' ' BUFFER OVERFLOW

1 MSALIN. 3 NSS ADJUSTED LINE NUMBER
1 NCHAN. 3 CHANNEL NUMBER
1 MSASLO. 3 LOW ADJUSTED SAMPLE NUMBER
1 MSASHI) 3 HIGH ADJUSTED SAMPLE NUMBER
-----
C
C
C
C HISTORY
C -----
C
C CHARLES HELMKE      LEC      07/25/79      REQUIREMENTS
C M A TOMPKINS      LEC      09/06/79      ALGORITHM DESIGN
C M A TOMPKINS      LEC      09/07/79      ALGORITHM CODING
C
C
C METHOD
C -----
C
C CHECK MSALIN <= 0 AND NHIBF<(PXBINS*5). IF TRUE RETURN. ELSE
C COMPUTE DIRECTION AND NUMBER OF RECORDS TO MOVE TAPE TO REQUESTED
C SCAN LINE AND CHANNEL. IF EOF IS ENCOUNTERED WHILE MOVING TAPE
C CALL ERTSHP FOR NEXT/PREVIOUS TAPE. READ RECORD INTO MPXBUF. SET:
C PXNOIN=129, PXNODA=129, PXLJOI=0, PXHJOI=0, PXBINT='BYT' AND REMAIN-
C DER OF BUFFER PREAMBLE POINTERS FOR THAT PART OF REQUESTED SAMPLE RANGE.
C IF ANY, WHICH IS INSIDE THE FILE.
C
C MACHINE-DEPENDENT CODE
C -----
C
C NONE.
C
C EXTERNAL REFERENCES
C -----
C
C ERTWAT 3 TIMED WAIT UP TO TO 30 SECONDS
C OCTQBY 3 OCT INTEGER FROM QUADRUPLE BYTE IN BYTE STRING
C MDEFATL 3 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C TSHAP3 3 SHAP TAPE
C PX4AM 3 PXBDEF PREAMBLE FOR MDP AM (UNCORRECTED NSS) BUFFER
C PX4PM 3 PXBDEF PREAMBLE FOR MDP PM (CORRECTED NSS) BUFFER
C PX4AR 3 PXBDEF PREAMBLE FOR MDP AR (UNCORRECTED RBV) BUFFER
C PX4PR 3 PXBDEF PREAMBLE FOR MDP PR (CORRECTED RBV) BUFFER
C MNOTE 3 PRINT/LOG 'NOTE' MESSAGES
C DOUBLE PRECISION COS4IN 3 VARIABLE-LENGTH (<=8 CHAR) STRING FOR INTEGER
C

```



ADJIL  
002

**N-298**

**DAN PACKAGE APPENDIX H  
UTILITY ROUTINES**

**R03BIL  
003**

```

C -----
C
C      MLOST=0
C      IOWAIT(LU3PKT)=0      & NO TIMED WAIT BETWEEN READS
C
C
C DETERMINE IF BUFFER IS ADEQUATE TO CONTAIN PREAMBLE
C
C      IF (MOD(MSALIN,50).EQ.0) CALL TRACE(CBS4IN(MSALIN,5))
C
C      ISTAT='OFL'
C      IF (NMIDP.LT.(PKBINS+5)) CALL MDPATL('MPKBUF TOO SMALL IN R03BIL')
C      IF (NMIDP.LT.(PKBINS+5)) GO TO 900
C
C      ISTAT='EOF'
C      IF (MSALIN.LE.0) CALL MDPATL('MSALIN <= 0 IN R03BIL')
C      IF (MSALIN.LE.0) GO TO 900
C
C
C COMPUTE RECORD AS FUNCTION OF INPUT SCAN LINE AND CHANNEL
C
C      MRAREC=(MSALIN-1)*NERCHA*NCHAN
C
C
C DETERMINE IF TAPE SWAP IS NECESSARY. NOTE: IN THE CASE THAT THE
C REQUESTED RECORD IS ON A PREVIOUS TAPE. THE REQUEST (I.E. TAPE SWAP)
C IS NOT HONORED UNLESS THE NUMBER OF RECORDS FOLLOWING THE REQUESTED
C RECORD IS GREATER THAN A SET TOLERANCE (TSRTOL).
C
C      150 IF ( MRAREC.LE.LU3RHI(LU3VOL) .AND. MRAREC.GE.LU3RLO(LU3VOL) )
C          & GO TO 240
C          IF ( MRAREC.GT.LU3RHI(LU3VOL) ) GO TO 200 & SWAP FORWARD
C          IF ( MRAREC.LT.LU3RLO(LU3VOL)-TSRTOL ) GO TO 200 & SWAP BACKWARD
C          GO TO 900
C
C LOCATE TAPE THAT CONTAINS RECORD
C
C      200 DO 220 I=1,5
C          LU3VOL=I
C          IF ( MRAREC.GE.LU3RLO(LU3VOL) .AND. MRAREC.LE.LU3RHI(LU3VOL) )
C              & GO TO 230
C      220 CONTINUE
C          CALL MDPATL('REQUESTED LINE'.CBS4IN(MSALIN,5)).
C          & ' NOT FOUND IN LU3BIL'
C          GO TO 900
C      230 CALL TSWAP3(ISTAT,LU3VOL)
C          IF (ISTAT.NE.' ') CALL MDPATL('TAPE CONTAINING LINE *'.
C          & CBS4IN(MSALIN,5)).' NOT ASSIGNED')
C          IF (ISTAT.NE.' ') GO TO 900
C
C
C COMPUTE DIRECTION AND NUMBER OF RECORDS TO MOVE TAPE
C
C      240 DO 400 I=1,2
C          MVREC=MRAREC-LU3RDF
C          IF (MVREC) 260,270,280

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**RD3BIL  
004**

```

C
C
C CASE OF MVREC NEG
C
200 MVREC=MVREC-1  0 RECORD BEFORE REQUESTED RECORD
    IF (MVDIR.NE.-1) CALL ERTHAT(2000)  0 REVERSE TAPES CAREFULLY
    MVDIR=-1
    GO TO 300

C
C
C CASE OF MVREC=0
C
270 MVREC=MVREC-1  0 RECORD BEFORE REQUESTED RECORD
    IF (MVDIR.NE.-1) CALL ERTHAT(2000)  0 REVERSE TAPES CAREFULLY
    MVDIR=-1
    GO TO 350

C
C
C CASE OF MVREC POS
C
200 CONTINUE
    IF (MVDIR.NE.1) CALL ERTHAT(2000)  0 REVERSE TAPES CAREFULLY
    MVDIR=1

C
C
C READ RECORD(S) UNTIL REQUESTED RECORD IS IN BUFFER
C
350 DO 450 MV = MVDIR,MVREC,MVDIR
    LU3RDF=LU3RDF+MVDIR
    CALL RTREC(MPXBUF(PXBINS),(NM10F-PXBINS+1),ISTAT, MVDIR,
    0 200,300,00)
    IF(ISTAT.EQ.'EOF') CALL HONOTE(
    1  ' UNEXPECTED EOF WHILE READING TO LINE 0'.
    2  COS4IN(MSALIN,0)')
    IF (ISTAT.EQ.'EOF') GO TO 470
    IF(ISTAT.EQ.'BADF') GO TO 000
450 CONTINUE
    IF (LU3RDF.EQ.MSAREC)GO TO 490
490 CONTINUE
    CALL HOFATL ('(LU3RDF<>MSAREC: SECOND ATTEMPT IN RD3BIL :')
    GO TO 000

C
C
C RECOVERY PROCEDURE FOR UNEXPECTED 'EOF'
C
470 IF (MVDIR.EQ.1) LU3RDF=LU3RMI(LU3VOL)
    IF (MVDIR.EQ.-1) LU3RDF=LU3RLO(LU3VOL)-1
    MVDIR=-MVDIR
    CALL RTREC(MPXBUF(PXBINS),(NM10F-PXBINS+1),ISTAT, MVDIR,
    0 200,300,00)
    MLOST=MLOST+1
    IF (MLOST.LE.2) GO TO 150
    ISTAT='EOF'
    GO TO 000
C

```

DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

RD3BIL  
005

```

C
C COMPUTE CURRENT LINE IN BUFFER
C
480 LU3LBF=(LU3RBF-1)/NERCHA + 1
C
C
C CHECK FOR NONRECOVERABLE READ STATUS
C
    IF (ISTAT.EQ.'LOST') CALL MNOTE(' (AT LINE *'.CBS4IN(MSALIN),
    & 'OPERATOR INDUCED LOST)' )
    IF (ISTAT.EQ.'LOST') ISTAT='BADF'
    IF (ISTAT.EQ.'BADF') GO TO 900
C
C
C OBTAIN/VERIFY TAPE TYPE AND RECORD NUMBER
C
    CALL GETBYT(NRECTY, MPXBUF(PXBINS),6)
    IF(NRECTY.EQ.0355) GO TO 485
    CALL MNOTE(' ( NON-IMAGE RECORD WHILE READING TO LINE *'.
    & CBS4IN(MSALIN,5),')')
    ISTAT = 'BADF'
    GO TO 900
485 CALL GETQBY(IREC, MPXBUF(PXBINS),1)
    IF (LU3RBF.EQ.IREC) GO TO 900 & RECORD IS REQUESTED RECORD
    IF (ISTAT.EQ.'BADR') GO TO 490
C
C
C RECOVERY PROCEDURE FOR 'LOST' WITH 'GOOD' STATUS
C
    LU3RBF=IREC & SET CURRENT RECORD IN BUFFER
    NLOST=NLOST+1
    IF (NLOST.LE.2) GO TO 150 & ATTEMPT TO LOCATE RECORD
    ISTAT='BADF'
    GO TO 900
C
C
C RECOVERY PROCEDURE FOR 'LOST' WITH 'BAD' STATUS
C
490 CALL TRECVR(ISTAT, MPXBUF(PXBINS),(NWBIF-PXBINS+1),MSAREC)
    IF (ISTAT.NE.' ') GO TO 900
    NLOST=NLOST+1
    IF (NLOST.LE.2) GO TO 150 & ATTEMPT TO LOCATE RECORD
    500 IF (ISTAT.EQ.'BADR') GO TO 900 & RECORD REQUESTED IS A BAD RECORD
C
C
C ESTABLISH PREAMBLE OF BUFFER
C
    IF (LU3REF(1).NE.'AM') GO TO 510
    CALL PX4AM(MPXBUF,ISTAT, MPXBUF,(NWBIF),NCHAN,MSASLO,MSASHI)
    GO TO 900
510 IF (LU3REF(1).NE.'PH') GO TO 520
    CALL PX4PH(MPXBUF,ISTAT, MPXBUF,(NWBIF),NCHAN,MSASLO,MSASHI)
    GO TO 900
520 IF (LU3REF(1).NE.'AR') GO TO 530
    CALL PX4AR(MPXBUF,ISTAT, MPXBUF,(NWBIF),NCHAN,MSASLO,MSASHI)
    GO TO 900

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**R03BIL  
006**

```
530 IF (L03REF(1).NE.'PR') GO TO 540
    CALL PX4PR(NPXBUF,ISTAT,  NPXBUF,(NHIBF),NCHAN,MSASLO,MSASHI)
    GO TO 900
540 ISTAT='BADF'
    CALL HDFATL ('UNSUPPORTED RECORD FORMAT IN R03BIL')
900 RETURN
END
```

DAH PACKAGE APPENDIX H  
UTILITY ROUTINES

RD3BIP  
001

```

SUBROUTINE RD3BIP( 0 READ (IF NOT IN BUFFER) FROM BIP TAPE TO PXBDEF FMT
0 MPXBUF. 0 NSS PIXEL BUFFER IN PXBDEF FORMAT
I NHIOF. 0 NUMBER OF WORDS IN I BUFFER
0 ISTAT. 0 I/O STATUS CODES:
C      .      .      .      .      .      .      .      .      .      .
C      'EOF'   END OF FILE OR REQUESTED DATA OUTSIDE FILE
C      'BADR'  BAD RECORD
C      'BADF'  BAD FILE
C      'OFL'   BUFFER OVERFLOW
C
C
I MSALIN. 0 NSS ADJUSTED LINE NUMBER
I NCHAN. 0 CHANNEL NUMBER
I NSASLO. 0 LOW ADJUSTED SAMPLE NUMBER
I NSASHI) 0 HIGH ADJUSTED SAMPLE NUMBER
C -----
C
C
C HISTORY
C -----
C
C      MARY TOMPKINS      LEC      07/23/79      REQUIREMENTS
C      MARY TOMPKINS      LEC      09/17/79      ALGORITHM DESIGN
C      MARY TOMPKINS      LEC      09/19/79      ALGORITHM CODING
C      MARY TOMPKINS      LEMSCO   06/12/80      STOP READING ON 'BADF'
C
C
C METHOD
C -----
C
C      CHECK MSALIN<=0 AND NHIOF<(PXBINS+5). IF TRUE RETURN. ELSE
C      COMPUTE DIRECTION AND NUMBER OF SCAN LINES TO MOVE TAPE
C      BLOCK(S) UNTIL REQUESTED LINE IS IN BUFFER. IF NECESSARY.
C      READ TAPE. UN-INTERLEAVE PIXEL DATA FOR DESIRED CHANNEL
C      INTO MPXBUF. SET: PXNODA = 129. PXNOIN = 128. PXLJOI = 0
C      PXHJOI = 0. PXBINT = 'BYT'. AND REMAINDER OF BUFFER PREAMBLE
C      POINTERS FOR THAT PART OF REQUESTED SAMPLE RANGE, IF ANY.
C      WHICH IS INSIDE THE FILE. UPDATE ISTAT AND PXQUAL AS REQUIRED.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE
C
C
C EXTERNAL REFERENCES
C -----
C
C      MOVCSF      0 MOVE CHARACTER STRING
C      ERWAIT      0 WAIT FOR COMPLETION OF I/O
C      ERTNAT      0 TIMED WAIT UP TO 30 SECONDS
C      ERIO        0 INITIATE I/O
C      MOFATL      0 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C      MOVDDBY     0 MOVE DOUBLE BYTE TO ONE STRING FROM ANOTHER
C      MNOTE       0 PRINT/LOG 'NOTE' MESSAGES
C      INTEGER NB4NI 0 NUMBER OF BYTES FOR NUMBER OF INTEGERS

```

**P.O. 381 P  
002**

**N-290**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

R03B1P  
003

```

C
C
      IF(MOD(MSALIN,50).EQ.0) CALL TRACE(CBSWIN(MSALIN,5))
C
C
C CHECK BUFFER SIZE
C
      ISTAT = 'OFL'
      IF(MNIBF.LT.(PXBINS+5)) CALL M0FATL('MPXBUF TOO SMALL IN R03B1P')
      IF(MNIBF.LT.(PXBINS+5)) GO TO 900
C
C
C CHECK FOR NEGATIVE OR ZERO LINE NUMBERS
C
      ISTAT = 'EOF'
      IF(MSALIN.LE.0) CALL M0FATL('MSALIN (= ZERO IN R03B1P')
      IF(MSALIN.LE.0) GO TO 900
C
C
C ON BIP TAPES LINE AND RECORD NUMBER ARE EQUAL
C
      MSAREC = MSALIN
C
C
C COMPUTE DIRECTION AND NUMBER OF RECORD(S) TO MOVE TAPE
C
      100 DO 300 I = 1,2
          MVREC = MSAREC - LU3RBF
          IF(MVREC)140,150,160
C
C CASE OF MVREC NEGATIVE
C
      140   MVREC = MVREC - 1      & RECORD BEFORE REQUESTED RECORD
          IF(MVDIR.NE.-1) CALL ERTWAT(2000) & REVERSE TAPES CAREFULLY
          MVDIR = - 1
          GO TO 200
C
C CASE OF RECORD ALREADY IN BUFFER
C
      150 CONTINUE
          IF(MVDIR.EQ.0) GO TO 140  & NEW OVERLAY----BUFFER EMPTY
          ISTAT = JSTAT  & USE STATUS OF INITIAL READ
          GO TO 300
C
C CASE OF MVREC POSITIVE
C
      160   IF(MVDIR.NE.+1) CALL ERTWAT(2000) & REVERSE TAPES CAREFULLY
          MVDIR = 1
C
C
C SET TIMED WAIT TO 10 MILLISECONDS BETWEEN CONSECUTIVE READS
C (THIS IS A FIX FOR A SYNCHRONIZATION BUG WITHIN THE UNIVAC-
C SUPPLIED TAPE I/O HANDLER ACCESSED THRU ER 10S/10WS/WAITS)
C
      200   IOWAIT(LU3PKT) = 0
          IF(MVREC.NE.1) IOWAIT(LU3PKT) = 10 & TIMED WAIT BETWEEN READS

```



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

RD3BIP  
884

```

C
C READ RECORD(S) UNTIL REQUESTED RECORD IS IN BUFFER
C
      DO 240 MV = MVDIR,MVREC,MVDIR
      LU3RBF = LU3RBF + MVDIR
      NIND = 1
      DO 230 NBLK = 1,LU3BIL
      CALL R3TREC(MSSBUF(NIND),(LU3HIB),ISTAT, MVDIR,200,
      300.00)
      JSTAT = ISTAT      & SAVE STATUS
      IF((ISTAT.EQ.'EOF').OR.(ISTAT.EQ.'BADF')) GO TO 300
230      NIND = NIND + LU3HIB
240      CONTINUE
      IF(MSAREC.EQ.LU3RBF) GO TO 300
300      CONTINUE
      CALL M0FATL( 'PROGRAMMING ERROR IN RD3BIP LOCATING RECORD')
      GO TO 900
C
C CHECK FOR BAD I/O STATUS
C
300      IF(ISTAT.EQ.'EOF') ISTAT = 'BADF'
      LU3LBF = LU3RBF
      IF(ISTAT.NE.' ') GO TO 900
C
C
C UN-INTERLEAVE PIXELS FROM MSSBUF AND MOVE TO MPXBUF
C
      NSMSSB = (MSA1HW(MSAM,WMAX)-MSA1HW(MSAM,WMIN)) + 1
      NSMPXB = NB4NI(NWIBF-PXBINS+1)
      NSNPAK = NIND(NSMSSB,NSMPXB)
      LOCOUT = 1      & BIN TO HOLD 1ST SAMPLE
      LOCIN = 2 + NCHAN - 1      & BIN CONTAINING FIRST SAMPLE MSSBUF
      DO 410 I = 1,NSNPAK,2
      CALL MOVB0Y(MPXBUF(PXBINS),LOCOUT, MSSBUF,LOCIN)
      LOCOUT = LOCOUT + 2
      LOCIN = LOCIN + 8
410      CONTINUE
C
      MPXBUF(PXRECIN) = LU3RBF      & INITIALIZE MPXBUF POINTER
C
      MPXLSA = MSA1HW(MSAM,WMIN)
      MPXHSA = NSNPAK + MPXLSA - 1
C
C
C CHECK IF SAMPLES REQUESTED LIE OUTSIDE DEFINED AREA
C
      IF( (MSASLO.GT.MPXHSA).OR.(MSASHI.LT.MPXLSA) ) GO TO 800
C
C
C INITIALIZE MPXBUF POINTERS
C
      MPXBUF(PXLSAM) = MAX0(MSASLO,MPXLSA)
      MPXBUF(PXHSA) = MIN0(MSASHI,MPXHSA)
      MPXBUF(PXLBIN) = MPXBUF(PXLSAM) - MPXLSA + 1
      MPXBUF(PXHBIN) = MPXBUF(PXHSA) - MPXLSA + 1
      MPXBUF(PXBINT) = 'BYT'

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**RD3B1P  
005**

**NPXBUF(PXQUAL) = 0  
NPXBUF(PXNOIN) = 128  
NPXBUF(PXNODA) = 129  
NPXBUF(PXLJOI) = 0  
NPXBUF(PXHJOI) = 0  
GO TO 900**

**C  
800 ISTAT = 'EOF'**

**C  
900 RETURN**

**C  
END**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

RO3880  
001

```

SUBROUTINE RO38SQ(      8 READ/CONVERT 8SQ FORMAT FROM TAPE INTO PXDEF
C                      FORMAT
C      0 MPXBUF.      8 MSS PIXEL BUFFER IN PXDEF FORMAT
C      1 NWIBF.      8 NUMBER OF WORDS IN 1 BUFFER
C      0 ISTAT.      8 I/O STATUS CODES
C                      .      . NORMAL COMPLETION
C                      'EOF' END OF FILE OR REQUESTED DATA
C                      OUTSIDE OF FILE
C                      'E10R' BAD RECORD
C                      'BADF' BAD FILE
C                      'OFL' BUFFER OVERFLOW
C
C
C      1 MSALIN.      8 MSS ADJUSTED LINE NUMBER
C      1 NCHAN.      8 CHANNEL NUMBER
C      1 MSASLO.      8 LOW ADJUSTED SAMPLE NUMBER
C      1 MSASHI      8 HIGH ADJUSTED
C
C -----
C
C HISTORY
C -----
C
C      MARY TOMPKINS      LEC      07/27/79      REQUIREMENTS
C                      LEC      / /      ALGORITHM DESIGN
C                      LEC      / /      ALGORITHM CODING
C
C
C METHOD
C -----
C
C      SET ISTAT = ' '. CHECK MSALIN (<= 0 AND NWIBF < (PXBSIN + 5). IF
C      ISTAT = ' ' COMPUTE DIRECTION AND NUMBER OF RECORDS TO MOVE TAPE
C      TO REQUESTED SCAN LINE WITHIN THE REQUESTED FILE. IF AN EOF
C      IS ENCOUNTERED WHILE READING FORWARD THEN CALL ERTSWP FOR THE
C      NEXT TAPE. OR AN EOF WHILE READING BACKWARD WITHIN THE FILE
C      CALL ERTSWP FOR THE PREVIOUS TAPE. READ RECORD INTO MPXBUF. SET
C      PXNOIN = 128. PXNODA = 129. PXLJOI = 0. PXHJOI = 0. PXBINT = 'BYT'
C      AND REMAINDER OF BUFFER PREAMBLE POINTERS FOR THAT PART OF
C      REQUESTED SAMPLE RANGE. IF ANY. WHICH IS INSIDE THE FILE.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      ERTWAT      8 TIMED WAIT UP TO TO 30 SECONDS
C      ERTSWP      8 SWAP TO REQUESTED TAPE
C      MOVCS      8 MOVE CHARACTER STRING
C      CST4IN      8 ENCODE CHARACTER STRING FROM INTEGER
C      MDFATL      8 PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

RD3B9Q  
002

```

C      TSNAP      & SNAP TAPE
C      PX4AH      & PXBDEF PREAMBLE FOR HOP AH (UNCORRECTED HSS) BUFFER
C      PX4PH      & PXBDEF PREAMBLE FOR HOP PH (CORRECTED HSS) BUFFER
C      PX4AR      & PXBDEF PREAMBLE FOR HOP AR (UNCORRECTED RBY) BUFFER
C      PX4PR      & PXBDEF PREAMBLE FOR HOP PR (CORRECTED RBY) BUFFER
C      MONOTE      & PRINT/LOG 'NOTE' MESSAGES
C
C EXCEPTIONS
C -----
C 1. THE FOLLOWING EXCEPTION CONDITIONS PRODUCE THE OUTPUTS SHOWN BELOW:
C
C      CONDITION          MPXBUF  MPXBUF  MPXBUF  MPXBUF  ISTAT  DIAONOSTIC
C                        (PXQUAL) (PXLSAH) (PXLSIN) (PXNOIN)
C                        (PXHSAM) (PXHSIN) (PXNODI)
C
C NHIBF<(PXBSIN+5)  UNDEFINED UNDEFINED UNDEFINED UNDEFINED 'OFL'  MOFATL
C MSALIN.LE.0      9          0          0          0          UNDEFINED 'EOF'  MOFATL
C LOST POSITION ('LOST') 9          0          0          0          UNDEFINED 'BAOR' QUEUED MONOTE
C                                     + LINE +
C BAD RECORD        9          0          0          0          UNDEFINED 'BADR' QUEUED MONOTE
C                                     + LINE +
C BAD FILE          9          0          0          0          UNDEFINED 'BADF' QUEUED MONOTE
C                                     + LINE +
C REQUESTED SAMPLES INSIDE 9          0          0          0          UNDEFINED 'EOF'  NONE
C FILE BUT OUTSIDE BUFFER
C NECESSARY TAPE NOT SPECI- 9          0          0          0          UNDEFINED 'EOF'  QUEUED MONOTE
C FIED ON ASSIGN CARD                                     + LINE +
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST  & COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
C      INCLUDE KOHLU3.LIST  & PACKET/POINTERS FOR UNIT 3
C      INCLUDE PXBDEF.LIST  & DEFINITION OF INTERNAL BUFFER STRUCTURE
C      INCLUDE KOMIO.LIST   & FORTRAN MANIPULATION OF ASSEMBLER I/O PACKETS
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER MPXBUF(NHIBF)  & ARGUMENT
C
C
C PROCEDURE
C
C      CALL MOFATL( 'RD3B9Q NOT YET IMPLEMENTED')
C
C 900 RETURN
C
C      END

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

RD3DSK  
001

```

SUBROUTINE RD3DSK(      8 READ 1 CHANNEL FOR 1 SCAN LINE IN PXDEF FORMAT
                        FROM DISK
C
C      0 MPXBUF.      8 MSS PIXEL BUFFER IN PXDEF FORMAT
C      1 NHIBF.      8 NUMBER OF WORDS IN 1 BUFFER
C      0 ISTAT.      8 I/O STATUS CODES
C                      .      . NORMAL COMPLETION
C                      'EOF' END OF FILE OR REQUESTED DATA
C                      OUTSIDE OF FILE
C                      'BADR' BAD RECORD
C                      'BADF' BAD FILE
C                      'OFL' BUFFER OVERFLOW
C
C
C      1 MSALIN.      8 MSS ADJUSTED LINE NUMBER
C      1 NCHAN.      8 CHANNEL NUMBER
C      1 MSASLO.      8 LOW ADJUSTED SAMPLE NUMBER
C      1 MSASHI      8 HIGH ADJUSTED SAMPLE NUMBER
C      -----
C
C HISTORY
C -----
C
C      MARY TOMPKINS      LEC      07/27/79      REQUIREMENTS
C      MARY TOMPKINS      LEC      08/10/79      STUBBED
C      E H SCHLOSSER      LEMSCO 06/20/80      ADAPTED FROM READ2N
C
C METHOD
C -----
C
C      SET ISTAT TO ' '. CHECK MSALIN <= 0 AND NHIBF < (PXBIN*5).
C      COMPUTE RECORD NUMBER FOR REQUESTED SCAN LINE AND CHANNEL.
C      READ INTO MPXBUF. IF PXQUAL = 9 SET ISTAT = 'BADR' AND RETURN
C      ELSE SET ISTAT EQUAL TO STATUS OBTAINED FROM DISK READ. IF
C      ISTAT <> ' ' SET PXQUAL = 9.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE
C
C EXTERNAL REFERENCES
C -----
C
C      MDPATL      8 PRINT/LOG/TALLY 'FATAL ERROR' MESSAGES
C      MDNOTE      8 PRINT AND LOG 'NOTE' DIAGNOSTIC MESSAGES
C      ERION      8 INITIATE I/O AND WAIT FOR COMPLETION
C      DOUBLE PRECISION COS4IN      8 VARIABLE LENGTH STRING FOR INTEGER
C      DOUBLE PRECISION COS4CS      8 VARIABLE LENGTH STRING FOR FIXED STRING
C
C
C EXCEPTIONS
C -----

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

RD3DSK  
002

```

C
C 1. THE FOLLOWING EXCEPTION CONDITIONS PRODUCE THE OUTPUTS SHOWN BELOW:
C
C CONDITION          MPXBUF      MPXBUF      MPXBUF      MPXBUF      ISTAT      DIAGNOSTIC
C                   (PXQUAL)    (PXLSAM)    (PXLSIN)    (PXNOIN)
C                   (PXHSAM)    (PXHSIN)    (PXNODA)
C
C NCHAN OUT OF RANGE      0          0          0          UNDEFINED 'EOF'  NDMARN
C MSASLO > MSASHI         0          0          0          UNDEFINED 'EOF'  MDMNOTE
C NO REQUESTED SAMPLES
C IN ANY STRIP            0          0          0          UNDEFINED 'EOF'  NONE
C BUFFER TOO SMALL
C TOO HOLD STRIP          0          0          0          UNDEFINED 'OFL'  MDMFATL
C BAD RECORD              0          0          0          UNDEFINED 'BADR'  QUEUED MDMNOTE
C                               + LINE NO
C BAD FILE                0          0          0          UNDEFINED 'BADF'  QUEUED MDMNOTE
C                               + LINE NO
C WRONG RECORD/LINE/CHAN
C RETRIEVED               0          0          0          UNDEFINED 'BADF'  MDMFATL
C INVALID BIN TYPE        0          0          0          UNDEFINED 'BADF'  MDMFATL
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
C      INCLUDE KOMLU3.LIST      & COMMON I/O PACKET/POINTERS FOR UNIT 3
C      INCLUDE KOMIO.LIST       & COMMON I/O FUNCTIONS
C      INCLUDE KOMNER.LIST      & COMMON ERTS SCENE PARAMETERS
C      INCLUDE KOMKLS.LIST      & COMMON CLASSIFICATION INFO
C      INCLUDE KOMFIT.LIST      & COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C      INCLUDE KOMIWM.LIST      & COMMON INPUT WINDOW PACKETS
C      INCLUDE WINDEF.LIST      & DEFINE STRUCTURE OF WINDOW PACKETS
C      INCLUDE PXBDEF.LIST      & DEFINE PIXEL BUFFER STRUCTURE
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER MPXBUF(NMIBF)    & ARGUMENT
C      INTEGER ISAMLO           & LOW SAMPLE REQUESTED
C      INTEGER ISAMHI           & HIGH SAMPLE REQUESTED
C      INTEGER NMHLO            & WORD IN MPXBUF TO BEGIN I/O AT
C      INTEGER NMHMI            & WORD IN MPXBUF FOR END OF READ BUFFER
C      INTEGER NREC             & RECORD NUMBER IN DISK FILE
C
C
C PROCEDURE
C -----
C
C      IF(MOD(MSALIN,50).EQ.0) CALL TRACE(CBS4IN(MSALIN,5))
C
C
C INITIALIZE STATUS TO EOF
C

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

RD3DSK  
063

```

      ISTAT = 'EOF'
C
C
C CHECK REQUESTED CHANNEL AND SAMPLE RANGE
C
      IF((NCHAN.LT.1).OR.
& (NCHAN.GT.NERCHA)) CALL MOWARN(
- 'BAD CHANNEL REQUESTED IN RD3DSK')
      IF(MSASLO.GT.MSASHI) CALL MNOTE('MSASLO > MSASHI IN RD3DSK')
      IF(MSASLO.GT.MSASHI) GO TO 830
C
      ISAMLO = MAX0(MSASLO,1) & LOW SAMPLE REQ.
      ISAMHI = MIN0(MSASHI,NERCHA) & HIGH SAM REQ.
C
C
C CHECK IF BUFFER WILL HOLD THIS STRIP
C
      NWOLO = 1 & WORD IN MPXBUF TO BEGIN I/O
      NMCHI = NWOLO + LU3LKS*20 - 1
      IF(NMCHI.LE.NHISF) GO TO 210
      CALL MOFATL('MPXBUF BUFFER PASSED TO RD3DSK TOO SMALL')
      ISTAT = 'OPL'
210      IF(INDTOTL.NE.0) GO TO 830
C
C
C COMPUTE RECORD NUMBER. READ INTO BUFFER AND CHECK I/O STATUS CODE
C
      NREC=(MSALIN-MSAIHW(WLIN.WHIN))*NERCHA+NCHAN
      CALL RD3REC
      IF(ISTAT.EQ.' ') GO TO 210
      IF(ISTAT.EQ.'BADR') CALL MNOTE(
- 'BAD RECORD AT LINE'.CBS4IN(MSALIN,6).')
      IF(ISTAT.EQ.'BADF') CALL MNOTE(
- 'BAD FILE AT LINE'.CBS4IN(MSALIN,6).')
      GO TO 830
C
C
C CHECK RECORD NUMBER AND LINE NUMBER IN BUFFER PREAMBLE
C
210      IF(MPXBUF(NWOLO+PXREC-1).NE.NREC) CALL MOFATL(
- 'RD3DSK READ RECORD'.
& CBS4IN(MPXBUF(NWOLO+PXREC-1),6). ' FOR'.CBS4IN(NREC,6))
C
      IF(MPXBUF(NWOLO+PXLIN-1).NE.MSALIN) CALL MOFATL(
- 'RD3DSK READ LINE'.
& CBS4IN(MPXBUF(NWOLO+PXLIN-1),6). ' FOR'.CBS4IN(MSALIN,6))
C
C
C CHECK CHANNEL NUMBER IN BUFFER PREAMBLE
C
      IF(MPXBUF(NWOLO+PXCHAN-1).NE.NCHAN) CALL MOFATL(
- 'RD3DSK READ CHANNEL'.
& CBS4IN(MPXBUF(NWOLO+PXCHAN-1),3). ' FOR'.CBS4IN(NCHAN,3))
      IF(INDTOTL.NE.0) GO TO 820
C
C

```

ROZOK  
004

6

**230** **CONTINUE**

5

**C**

**C ADJUST PREAMBLE POINTERS TO REFLECT REQUESTED DATA ONLY**

**C**

**C**

**C**

C STATUS IS BAD FILE

C

020 ISTAT - 'BADF'

**C**

**C**

**C SET PREAMBLE TO NO DATA**

**C**

6

**C**

**100 RETURN**

3

3

2

## 2

2

2

2

## INTERNAL SUBROUTINE R3DREC

3

3



**BAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**RD3D9K  
005**

**IOSIZE(LU3PKT) = LU3LRS\*20  
IOADDR(LU3PKT) = LOC(INPXBUF(MHOLE))  
IOSECT(LU3PKT) = 10\*LU3LRS\*(NREC-1)  
IOFUNC(LU3PKT) = 'BK' & READ  
CALL ERION(LU3PKT)  
ISTAT = IOCODE(LU3PKT)**

**C**

**RETURN**

**C**

**END**

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

RO3NUL  
001

```

SUBROUTINE RO3NUL(      & SIMULATES DATA AND CONSTRUCTS NULL BUFFER
                        IN PXDEF FORMAT
C
C 0 MPXBUF.      & NSS PIXEL BUFFER IN PXDEF FORMAT
C 1 NWIBF.      & NUMBER OF WORDS IN BUFFER
C 0 ISTAT.      & I/O STATUS CODES:
C                ' ' NORMAL COMPLETION
C                'OFL' BUFFER OVERFLOW
C
C
C
C 1 MSALIN.      & NSS ADJUSTED LINE NUMBER
C 1 NCHAN.      & CHANNEL NUMBER
C 1 MSASL.      & LOW ADJUSTED SAMPLE NUMBER
C 1 MSASHI.      & HIGH ADJUSTED SAMPLE NUMBER
C -----
C
C
C HISTORY
C -----
C
C MARY TOMPKINS      LEC      07/24/79      REQUIREMENTS
C MARY TOMPKINS      LEC      09/12/79      ALGORITHM DESIGN
C MARY TOMPKINS      LEC      09/12/79      ALGORITHM CODING
C
C
C METHOD
C -----
C
C CHECK MSALIN<=0 AND NWIBF<(PXBSIN+5). IF TRUE RETURN.ELSE
C GENERATE SIMULATED PIXEL DATA STORE IN PXDEF FORMAT. SET:
C PXNOIN = 129. PXNODA = 129. PXLJOI = 0. PXHJOI = 0
C PXBINT = 'BYT'.PXQUAL = 0. AND REMAINDER OF BUFFER PREAMBLE
C POINTERS FOR THAT PART OF REQUESTED SAMPLE RANGE IF ANY
C GREATER THAN ZERO AND LESS THAN 5000.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C NONE
C
C EXTERNAL REFERENCES
C -----
C
C PUTBYT      & PUT ONE NON-NEG INTEGER INTO BYTE STRING
C MDEFATL      & PRINT/LOG/COUNT 'FATAL ERROR' MESSAGES
C INTEGER NB4NI      & # OF BYTES FOR # OF INTEGERS
C
C
C EXCEPTIONS
C -----
C
C 1. THE FOLLOWING EXCEPTION CONDITIONS PRODUCE THE OUTPUT SHOWN BELOW:
C CONDITION      MPXBUF      MPXBUF      MPXBUF      MPXBUF      ISTAT      DIAGNOSTIC
C                (PXQUAL) (PXLSAM) (PXLBIN) (PXNOIN)
C                (PXHSAM) (PXHBIN) (PXNODA)

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

RD3NUL  
002

```

C NHIBF<(PXBINS+5) UNDEFINED UNDEFINED UNDEFINED UNDEFINED 'EOF' M0FATL
C MSALIN<=0          9          0          0          UNDEFINED 'EOF' NONE
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KONXQT.LIST      3 COMMON PROGRAM EXECUTION SWITCHES.COUNTERS
C      INCLUDE PXBDEF.LIST      3 DEFINITION OF INTERNAL BUFFER STRUCTURE
C      INCLUDE KOHLU3.LIST      3 PACKET/POINTERS FOR UNIT 3
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER MPXBUF(NHIBF)    3 ARGUMENT
C      INTEGER NSIPIX           3 NUMBER OF SIMULATED PIXELS
C
C
C PROCEDURE
C -----
C
C      CALL TRACE
C
C
C CHECK BUFFER LENGTH
C
C      ISTAT = 'OFL'
C      IF(NHIBF.LT.(PXBINS+5)) CALL M0FATL('BUFFER TOO SMALL IN RD3NUL')
C      IF(NHIBF.LT.(PXBINS+5)) GO TO 900
C
C      ISTAT = 'EOF'
C      IF(MSALIN.LE.0) CALL M0FATL('MSALIN <= 0 IN RD3NUL')
C      IF(MSALIN.LE.0) GO TO 900
C
C
C SIMULATE DATA FOR REQUESTED NUMBER OF SAMPLES FOR BUFFER LENGTH
C
C      NSIPIX = MIN0(NB4NI(NHIBF-PXBINS+1),MSASHI-MSASLO+1)
C      DO 100 NPIX = 1,NSIPIX
C          CALL PUTBYT(MPXBUF(PXBINS),(NPIX), NCHAN)
C      100 CONTINUE
C
C
C INITIALIZE BUFFER POINTERS
C
C      MPXBUF(PXBINT) = 'BYT'
C      MPXBUF(PXQUAL) = 0
C      MPXBUF(PXLSAN) = MSASLO
C      MPXBUF(PXHSAH) = MSASLO + NSIPIX - 1
C      MPXBUF(PXLBIN) = 1
C      MPXBUF(PXHBIN) = NSIPIX
C      MPXBUF(PXNOIN) = 128
C      MPXBUF(PXNODI) = 129
C      MPXBUF(PXLJOI) = 0
C      MPXBUF(PXHJOI) = 0

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

RD3HVL  
003

ISTAT = . . .  
C  
900 RETURN  
C  
END

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

READ2N  
001

```

SUBROUTINE READ2N( 3 READ ONE CHANNEL FOR 1 SCAN LINE FROM
DETECTION FILE(S)
C
O MPXBUF. 3 MSS PIXEL BUFFER IN PXBDEF FORMAT
I NMIBF. 3 NUMBER OF WORDS IN 1 BUFFER
O ISTAT. 3 I/O STATUS CODE      *NORMAL COMPLETION
C                                     *EOF * END OF FILE OR REQUESTED DATA
C                                     OUT OF FILE
C                                     *BADR* BAD RECORD
C                                     *BADF* BAD FILE
C                                     *OFL* BUFFER OVERFLOW
C
I MSALIN. 3 MSS ADJUSTED LINE NUMBER
I NCHAN. 3 CHANNEL NUMBER
I MSASLO. 3 LOW ADJUSTED SAMPLE NUMBER
I MSASHI) 3 HIGH ADJUSTED SAMPLE NUMBER
-----
C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER LEC      07/12/79      REQUIREMENTS
C      CHARLES HELMKE LEC     12/18/79      ALGORITHM DESIGN
C      CHARLES HELMKE LEC     12/19/79      ALGORITHM CODING
C      E H SCHLOSSER LEMSCO 05/16/80      SUPPORT MULTIPLE DETECTION CHANNELS
C
C
C METHOD
C -----
C
C      CHECK/FLAG DETECTION RECORD AVAILABILITY.READ REQUIRED
C      RECORD(S) FROM DETECTION FILE(S).JOIN THEM. AND RETURN
C      REQUESTED DETECTION DATA FOR LINE MSALIN.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      MDPATL 3 PRINT/LOG/TALLY 'FATAL ERROR' MESSAGES
C      MDNOTE 3 PRINT AND LOG 'NOTE' DIAGNOSTIC MESSAGES
C      JOIN2N 3 JOIN DENSITY DET REC AND CONCATENATE
C      ERION 3 INITIATE I/O AND WAIT FOR COMPLETION
C      INTEGER NI4NB 3 NUMBER OF INTEGERS FOR NUMBER OF BYTES
C      INTEGER NI4NC 3 NUMBER OF INTEGERS FOR NUMBER OF CHARACTERS
C      DOUBLE PRECISION CBS4IN 3 VARIABLE LENGTH STRING FOR INTEGER
C      DOUBLE PRECISION CBS4CS 3 VARIABLE LENGTH STRING FOR FIXED STRING
C
C
C EXCEPTIONS
C -----

```

```

C
C 1. THE FOLLOWING EXCEPTION CONDITIONS PRODUCE THE OUTPUTS SHOWN BELOW:
C
C      CONDITION      MPXBUF      MPXBUF      MPXBUF      MPXBUF      ISTAT      DIAGNOSTIC
C                    (PXQUAL) (PXLSAM) (PXLBIN) (PXNOIN)
C                    (PXMSAM) (PXMBIN) (PXNODI)
C
C NCHAN OUT OF RANGE  9          0          0          UNDEFINED 'EOF'  MDWARN
C MSASLO > MSASHI    9          0          0          UNDEFINED 'EOF'  MDNOTE
C NO REQUESTED SAMPLES
C IN ANY STRIP      9          0          0          UNDEFINED 'EOF'  NONE
C BUFFER TOO SMALL
C TOO MANY STRIP(S)  9          0          0          UNDEFINED 'OFL'  MDEFATL
C BAD RECORD        9          0          0          UNDEFINED 'BAOR'  QUEUED MDNOTE
C                                     + LINE NO
C BAD FILE          9          0          0          UNDEFINED 'BAOF'  QUEUED MDNOTE
C                                     + LINE NO
C WRONG RECORD/LINE/CHAN
C RETRIEVED         9          0          0          UNDEFINED 'BAOF'  MDEFATL
C INVALID BIN TYPE   9          0          0          UNDEFINED 'BAOF'  MDEFATL
C ERROR FROM JOIN2N  9          0          0          UNDEFINED 'BAOF'  MDEFATL
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST      2 COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
C      INCLUDE KOHL2N.LIST      2 COMMON I/O PACKETS FOR DETECTION FILES (21-24)
C      INCLUDE KOHIO.LIST       2 COMMON I/O FUNCTIONS
C      INCLUDE KOMNER.LIST      2 COMMON ERTS SCENE PARAMETERS
C      INCLUDE KOMKLS.LIST      2 COMMON CLASSIFICATION INFO
C      INCLUDE KOMFIT.LIST      2 COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C      INCLUDE KONDET.LIST      2 COMMON DETECTION FILE WINDOWS
C      INCLUDE WINDEF.LIST      2 DEFINE STRUCTURE OF WINDOW PACKETS
C      INCLUDE PXBDEF.LIST      2 DEFINE BUFFER STRUCTURE
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER MPXBUF(NWIBF)    2 ARGUMENT
C      INTEGER ISAMLO           2 MSASLO-1
C      INTEGER ISAMHI           2 MSASHI-1
C      INTEGER NCCTLO           2 LOWEST STRIP NEEDED TO SUPPLY REQUESTED DATA
C      INTEGER NCCTHI           2 HIGHEST STRIP NEEDED TO SUPPLY REQUESTED DATA
C      INTEGER NWDLO            2 WORD IN MPXBUF TO BEGIN I/O AT
C      INTEGER NWDHI            2 WORD IN MPXBUF FOR END OF READ BUFFER
C      INTEGER NBINS            2 NO OF HIGHEST BIN USED
C      INTEGER NWDBIN           2 NO OF WORDS REQUIRED TO ACCOMMODATE BINS
C      INTEGER NREC             2 RECORD NUMBER IN DETECTION FILE
C
C
C PROCEDURE
C -----
C
C

```

DAM PACKAGE APPENDIX H  
UTILITY ROUTINES

READ2N  
003

```

        IF(MOD(MSALIN,50).EQ.0) CALL TRACE(CBS*IN(MSALIN,5))
C
C
C INITIALIZE STATUS TO EOF
C
        ISTAT = 'EOF'
C
C
C CHECK REQUESTED CHANNEL AND SAMPLE RANGE
C
        IF((NCHAN.LT.1).OR.
        & (NCHAN.GT.NERCHA)) CALL HDWARN(
        & 'BAD DETECTION CHANNEL IN READ2N')
        IF(MSASLO.GT.MSASHI) CALL HDNOTE('MSASLO > MSASHI IN READ2N')
        IF(MSASLO.GT.MSASHI) GO TO 830
C
        ISAMLO = MAX0(MSASLO-1,1) & LOW SAMPLE REQ. FOR JOIN MSASLO-1
        ISAMHI = MIN0(MSASHI+1, NERSAM) & HIGH SAM REQ. FOR JOIN MSASHI+1
C
C
C DETERMINE LEFTMOST AND RIGHTMOST STRIPS NEEDED
C
        NCCTLO = 0
        NCCTHI = 0
        DO 200 I=1,4
            IF(MSALIN.GT.MSADWH(WLIN,WMAX,1)) GO TO 200
            IF(MSALIN.LT.MSADWH(WLIN,WMIN,1)) GO TO 200
            IF(ISAMLO.GT.MSADWH(WSAM,WMAX,1)) GO TO 200
            IF(ISAMHI.LT.MSADWH(WSAM,WMIN,1)) GO TO 200
            IF(NCCTLO.EQ.0) NCCTLO=I
            NCCTHI = I
        200 CONTINUE
        IF(NCCTLO.EQ.0) GO TO 830 & NO DATA FOR THIS LINE
C
C
C LOOP TO BUILD BUFFER FROM RECORDS IN DETECTION
C FILES FOR DIFFERENT CCT STRIPS
C
        NWOLO = 1 & WORD IN MPXBUF TO BEGIN I/O
        DO 300 NCCT = NCCTLO,NCCTHI
C
C
C CHECK IF BUFFER WILL HOLD THIS STRIP
C
        NWOHI = NWOLO + LOETRS(NCCT)*28 - 1
        IF(NWOHI.LE.NHIBF) GO TO 210
        CALL HDFAIL('READ2N DETECTS BUFFER OVERFLOW')
        ISTAT = 'OFL'
        210 IF(NDOTOL.NE.0) GO TO 830
C
C
C COMPUTE RECORD NUMBER. READ INTO BUFFER AND CHECK I/O STATUS CODE
C
        NREC=(MSALIN-MSADWH(WLIN,WMIN,NCCT))*NERCHA+NCHAN
        CALL READET(LENPKT(1,NCCT))
        IF(ISTAT.EQ.'') GO TO 218

```

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

READ2N  
004

```

        IF(ISTAT.EQ.'BADR') CALL HDNOTE(
        * (BAD RECORD AT LINE*.CBS4IN(MSALIN.6).
        * OF STRIP*.CBS4IN(NCCT.2).') )
        IF(ISTAT.EQ.'BADF') CALL HDNOTE(
        * (BAD FILE AT LINE*.CBS4IN(MSALIN.6).
        * OF STRIP*.CBS4IN(NCCT.2).') )
        GO TO 030

C
C
C CHECK RECORD NUMBER AND LINE NUMBER IN BUFFER PREAMBLE
C
210      IF(MPXBUF(NWDOLO+PXRECN-1).NE.NREC) CALL HDFATL(
        * 'READ2N READ RECORD'.
        * CBS4IN(MPXBUF(NWDOLO+PXRECN-1).6). ' FOR'.CBS4IN(NREC.6).
        * ON STRIP*.CBS4IN(NCCT.2))
C
        IF(MPXBUF(NWDOLO+PXLINO-1).NE.MSALIN) CALL HDFATL(
        * 'READ2N READ LINE'.
        * CBS4IN(MPXBUF(NWDOLO+PXLINO-1).6). ' FOR'.CBS4IN(MSALIN.6).
        * ON STRIP*.CBS4IN(NCCT.2))
C
C
C CHECK CHANNEL NUMBER IN BUFFER PREAMBLE
C
        IF(MPXBUF(NWDOLO+PXCHAN-1).NE.NCHAN) CALL HDFATL(
        * 'READ2N READ CHANNEL'.
        * CBS4IN(MPXBUF(NWDOLO+PXCHAN-1).3). ' FOR'.CBS4IN(NCHAN.3).
        * ON STRIP*.CBS4IN(NCCT.2))
        IF(NDTOTL.NE.0) GO TO 020
C
C
C CHECK BIN TYPE IN BUFFER PREAMBLE
C
        IF(MPXBUF(NWDOLO+PXBINT-1).EQ.'BYT') GO TO 230
        IF(MPXBUF(NWDOLO+PXBINT-1).EQ.'CHR') GO TO 230
        IF(MPXBUF(NWDOLO+PXBINT-1).EQ.'INT') GO TO 230
        CALL HDFATL( 'READ2N DETECTS INVALID BIN TYPE'.
        * CBS4CS(MPXBUF(NWDOLO+PXBINT-1).6). ' ON LINE'.
        * CBS4IN(MSALIN.6). ' STRIP*.CBS4IN(NCCT.2))
        GO TO 020
C
C
C IF NOT LOWEST STRIP, THEN JOIN CURRENT STRIP TO PREVIOUS ONE(S)
C
230      IF(NCCT.EQ.NCCTLO) GO TO 240
        CALL JOIN2N(MPXBUF(1).JSTAT. MPXBUF(1).MPXBUF(NWDOLO))
        IF(JSTAT.NE.' ') GO TO 020
C
C
C IF MORE STRIPS, THEN UPDATE NWDOLO (WD TO BEGIN I/O FOR NEXT STRIP AT)
C
240      IF(NCCT.EQ.NCCTHI) GO TO 300
        NBINS = MPXBUF(PXHBIN)
        NWDBIN = NBINS
        IF(MPXBUF(PXBINT).EQ.'BYT') NWDBIN = N14NB(NBINS)
        IF(MPXBUF(PXBINT).EQ.'CHR') NWDBIN = N14NC(NBINS)

```



READEN  
008

**N-316**

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**READEN  
008**

```
      CALL ERION(NPKT)
      ISTAT = ICODE(NPKT)
C
      RETURN
C
      END
```

```

SUBROUTINE READ3:  & READ ONE RAW CHANNEL FOR PART OF ONE SCAN LINE
                   FROM LOGICAL UNIT 3
C
O MPXBUF,  & MSS PIXEL BUFFER IN PXBDEF FORMAT
I NHISF,  & NUMBER OF WORDS IN I BUFFER
O ISTAT,  & I/O STATUS CODE      '      ' NORMAL COMPLETION
C                                     'EOF' END OF FILE OR REQUESTED DATA
C                                     OUT OF FILE
C                                     'BADR' BAD RECORD
C                                     'BADF' BAD FILE
C                                     'OFL' BUFFER OVERFLOW
C
I MSALIN,  & MSS ADJUSTED LINE NUMBER
I NCHAN,  & CHANNEL NUMBER
I MSASLO,  & LOW ADJUSTED SAMPLE NUMBER
I MSASHI,  & HIGH ADJUSTED SAMPLE NUMBER
-----
C
C HISTORY
C -----
C
C      ED SCHLOSSER   LEC   07/12/79   REQUIREMENTS
C      MARY TOMPKINS  LEC   09/10/79   ALGORITHM DESIGN
C      MARY TOMPKINS  LEC   09/11/79   ALGORITHM CODING
C
C METHOD
C -----
C
C      IF NHISF IS ZERO, POSITION TO MAXO(MSALIN-1,1). ELSE INSERT MSALIN
C      AND NCHAN INTO MPXBUF(PXLINO) AND MPXBUF(PXCHAN). CHECK/FLAG TOO
C      FEW BINS IN BUFFER, MSALIN NOT IN ENVELOPE, NCHAN NOT IN FILE.
C      MSASLO>MSASHI. IF TRUE RETURN, ELSE CALL RD3... DEPENDING ON LU3SEQ.
C      ON RETURN FROM RD3... MPXBUF(PXLINO), MPXBUF(PXCHAN) WILL BE UN-
C      CHANGED AND REMAINDER OF BUFFER WILL CONTAIN POINTERS, FLAGS, AND
C      DATA FOR THAT PART OF REQUESTED SAMPLE RANGE, IF ANY, WHICH IS INSIDE
C      THE FILE. IF ISTAT IS 'BADR' ADD 1 TO LU3CBR, ELSE IF 'BADF' LU3CBR
C      = LU3HBR, ELSE LU3CBR=0.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      RD3BIP  & READ/CONVERT/UN-INTERLEAVE BIP FORMAT TO PXBDEF FORMAT
C      RD3BIL  & READ/CONVERT CIL FORMAT FROM TAPE INTO PXBDEF FORMAT
C      RD3BSQ  & READ/CONVERT BSQ FORMAT FROM TAPE INTO PXBDEF FORMAT
C      RD3DSK  & READ PXBDEF FORMAT FROM DISK
C      RD3NUL  & GENERATE TEST DATA IN PXBDEF FORMAT
C      NDFATL  & PRINT/LOG/COUNT 'FATAL ERROR' MESSAGE
C      NDNOTE  & PRINT/LOG 'NOTE' MESSAGE

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

READS  
002

```

DOUBLE PRECISION CBS4IN      8 VARIABLE-LENGTH STRING FOR INTEGER
C
C
C GLOBAL DECLARATIONS
C -----
C
    INCLUDE KONXQT.LIST      8 COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
    INCLUDE KONNER.LIST      8 COMMON ERTS SCENE PARAMETERS
    INCLUDE KONLUS.LIST      8 I/O AND UNPACKING DATA FOR MSS/RBY ON UNIT 3
    INCLUDE MINDEF.LIST      8 DEFINITION OF WINDOW PACKET STRUCTURE
    INCLUDE KONIMW.LIST      8 INPUT WINDOW PACKET
    INCLUDE PXBDEF.LIST      8 DEFINITION OF INTERNAL BUFFER STRUCTURE
C
C
C LOCAL DECLARATIONS
C -----
C
    PARAMETER DUMSIZ=PXBSIZ*5
    INTEGER MPXBUF(MNIBF)      8 ARGUMENT
    INTEGER MPXDUM(DUMSIZ)      8 DUMMY BUFFER USED FOR POSITIONING TAPE
C
C
C EXCEPTIONS
C -----
C
    1. THE FOLLOWING EXCEPTION CONDITIONS PRODUCE THE OUTPUTS SHOWN BELOW:
C


| CONDITION                  | MPXBUF<br>(PXQUAL) | MPXBUF<br>(PXLSAM) | MPXBUF<br>(PXLBIN) | MPXBUF<br>(PXNOIN) | MPXBUF<br>(PXMSAM) | MPXBUF<br>(PXMBIN) | MPXBUF<br>(PXNODI) | ISTAT            | DIAGNOSTIC                |
|----------------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|--------------------|------------------|---------------------------|
| * BINS IN BUF < UNDEFINED  | UNDEFINED          | UNDEFINED          | UNDEFINED          | UNDEFINED          | UNDEFINED          | UNDEFINED          | UNDEFINED          | 'OFL'            | NOFATL                    |
| (MSASHI-MSASLO+1)          |                    |                    |                    |                    |                    |                    |                    |                  |                           |
| MSALIN OUTSIDE INPUT 8     | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | UNDEFINED 'EOF'  | NONE                      |
| ENVELOPE                   |                    |                    |                    |                    |                    |                    |                    |                  |                           |
| NCHAN INVALID 8            | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | UNDEFINED 'EOF'  | MONOTE                    |
| NO DATA FOR NCHAN 8        | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | UNDEFINED 'EOF'  | NOFATL                    |
| MSASLO > MSASHI 8          | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | UNDEFINED 'EOF'  | MONOTE                    |
| NO REQUESTED SAMPLES 8     | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | UNDEFINED 'EOF'  | NONE                      |
| INSIDE INPUT ENVELOPE/FILE |                    |                    |                    |                    |                    |                    |                    |                  |                           |
| BAD RECORD 8               | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | UNDEFINED 'BADR' | QUEUED MONOTE<br>* LINE * |
| LUNBR CONSECUTIVE 8        | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | UNDEFINED 'BADF' | NONE                      |
| 'BADR'S                    |                    |                    |                    |                    |                    |                    |                    |                  |                           |
| BAD FILE 8                 | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | 0                  | UNDEFINED 'BADF' | QUEUED MONOTE<br>* LINE * |
| SHORT BLOCK IGNORED NA     | NA                 | NA                 | NA                 | NA                 | NA                 | NA                 | NA                 | NA               | QUEUED MONOTE<br>* LINE * |


C
C
C 2. IF MNIBF=0, THIS DOES NOT GENERATE AN 'OFL' STATUS. INSTEAD, IT
C RESULTS IN THE APPROPRIATE NORMAL OR EXECUTION STATUS, BUT NO OUTPUT
C TO EITHER THE PREAMBLE OR BINS OF MPXBUF. THIS ALLOWS POSITIONING
C AN EXTERNAL DEVICE TO MSALIN WITHOUT ACTUALLY RETURNING ITS CONTENTS.
C
C
C PROCEDURE
C -----
C

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

READ3  
002

```

        IF (MOD(MSALIN,50).EQ.0) CALL TRACE(CBS4IN(MSALIN,5))
C
        ISTAT=' '
C
C
C POSITION BUFFER SIZE IS ZERO
C
        IF (NMIBF.EQ.0) NCHAN=1
        IF (NMIBF.EQ.0 .AND. LUSSEQ(1).EQ.'BIP') CALL RD3BIP (
1 MPXDUM,(DUMSZ),ISTAT,MAX0(MSALIN-1),NCHAN,MSASLO,MSASHI)
        IF (NMIBF.EQ.0 .AND. LUSSEQ(1).EQ.'BIL') CALL RD3BIL (
1 MPXDUM,(DUMSZ),ISTAT,MAX0(MSALIN-1),NCHAN,MSASLO,MSASHI)
        IF (NMIBF.EQ.0 .AND. LUSSEQ(1).EQ.'BSQ') CALL RD3BSQ (
1 MPXDUM,(DUMSZ),ISTAT,MAX0(MSALIN-1),NCHAN,MSASLO,MSASHI)
        IF (NMIBF.EQ.0) GO TO 900
C
C
C CHECK IF BUFFER WILL HOLD NUMBER OF REQUESTED SAMPLES
C
        ISTAT='OFL'
        IF (NB4NI(NMIBF-PXBINS+1) .LE. (MSASHI-MSASLO+1)) GO TO 900
C
C
C INITIALIZE BUFFER PREAMBLE TO DEFAULT VALUES
C
        MPXBUF(PXL'NO')=MSALIN
        MPXBUF(PXCHAN)=NCHAN
        MPXBUF(PXBINT)='NUL'
        MPXBUF(PXQUAL)=0
        MPXBUF(PXLSAM)=0
        MPXBUF(PXHSAM)=0
        MPXBUF(PXLJOI)=0
        MPXBUF(PXHJOI)=0
        MPXBUF(PXLBIN)=0
        MPXBUF(PXHBIN)=0
        ISTAT='EOF'
C
C
C CHECKS TO DETERMINE IF REQUESTED DATA IS AVAILABLE --
C
C
        IF (LUSSEQ(1).EQ.'NUL') GO TO 300 & NO CHECK IF NUL DATA
C
C
C --ARE REQUESTED LINES OR SAMPLES TOTALLY OUTSIDE INPUT ENVELOPE/FILE ?
C
        IF ((MSALIN.LT.MSAIMH/(MLIN.MMIN)) .OR.
& (MSALIN.GT.MSAIMH/(MLIN.MMAX))) GO TO 900
        IF ((MSASHI.LT.MSAIMH/(MSAM.MMIN)) .OR.
& (MSASLO.GT.MSAIMH/(MSAM.MMAX))) GO TO 900
C
C
C --IS REQUESTED LOW SAMPLE > HIGH SAMPLE ?
C
        IF (MSASLO.GT. MSASHI) CALL MNOTE('MSASLO > MSASHI IN READ3')
        IF (MSASLO.GT.MSASHI) GO TO 900

```

BAN PACKAGE APPENDIX M  
UTILITY ROUTINES

READS  
000

```

C
C
C --IS CHANNEL OUTSIDE ALLOWED RANGE ?
C
    IF (NCHAN.LT.1 .OR. NCHAN.GT.NERCHA) CALL MONOTE (CBS4IN(NCHAN,3),
    1 'NOT WITHIN VALID RANGE FOR CHANNEL NUMBER')
    IF (NCHAN.LT.1 .OR. NCHAN.GT.NERCHA) GO TO 900

C
C
C --IS DATA AVAILABLE FOR REQUESTED CHANNEL ?
C
    IF (NERBAN(NCHAN).EQ.' ' .AND. LUJSEQ(1).NE.'NUL') CALL MDPATL(
    1 'CHANNEL'.CBS4IN(NCHAN,2). 'NOT AVAILABLE FROM REEL *',
    2 CBS4CS(LUJREL(LUJVOL),1,6))
    IF (NERBAN(NCHAN).EQ.' ' .AND. LUJSEQ(1).NE.'NUL') GO TO 900

C
C CALL APPROPRIATE RD... FOR I/O
C
    300 IF(LUJSEQ(1).EQ.'NUL')CALL RD3NUL(MPXBUF,(NHIBF),ISTAT,    MSALIN,
    1 NCHAN,MSASLO,MSASHI)
    IF(LUJSEQ(1).EQ.'BIP')CALL RD3BIP(MPXBUF,(NHIBF),ISTAT,    MSALIN,
    1 NCHAN,MSASLO,MSASHI)
    IF(LUJSEQ(1).EQ.'BIL')CALL RD3BIL(MPXBUF,(NHIBF),ISTAT,    MSALIN,
    1 NCHAN,MSASLO,MSASHI)
    IF(LUJSEQ(1).EQ.'BSQ')CALL RD3BIL(MPXBUF,(NHIBF),ISTAT,    MSALIN,
    1 NCHAN,MSASLO,MSASHI)
    IF(LUJSEQ(1).EQ.'DSK')CALL RD3DSK(MPXBUF,(NHIBF),ISTAT,    MSALIN,
    1 NCHAN,MSASLO,MSASHI)

C
C CHECK I/O STATUS
C
    IF (ISTAT.EQ.' ')LUJCBR=0
    IF (ISTAT.EQ.' ')GO TO 900
    IF (ISTAT.NE.'BADR')GO TO 900
    CALL MONOTE ( 'BAD RECORD AT LINE *'.CBS4IN(MSALIN,5). ' ')
    LUJCBR=LUJCBR+1
    IF (LUJCBR.LT.LUJNBR) GO TO 900
    ISTAT='BADF'
    GO TO 900
    600 IF(ISTAT.NE.'BADF') GO TO 900
    CALL MONOTE ( 'BAD FILE AT LINE *'.CBS4IN(MSALIN,5). ' ')

C
C
C 900 RETURN
C
    END

```

**READS  
001**

```

C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      03/08/78      ORIGINAL CODE IN READS & BLOCKS
C      E M SCHLOSSER      LEC      02/14/78      REVISE & CONSOLIDATE
C      E M SCHLOSSER      LEC      05/01/79      DON'T PROMPT WITHIN ADD
C      E M SCHLOSSER      LEC      10/16/79      DE-QUEUE QUEUED DIAGNOSTIC MESSAGES
C
C
C METHOD
C -----
C
C THIS SUBROUTINE MAINTAINS THE IMAGE BUFFER FOR UNIT 5 (CARD READER
C OR DEMAND TERMINAL) FOR THE DAM PACKAGE.
C
C
C FREE-FORMAT INPUT RULES
C -----
C
C 1. RECORDS -- AN INPUT RECORD CONTAINS UP TO 80 COLUMNS ENDING WITH
C    COLUMN 80 (CARD READER) OR TERMINATED BY A CARRIAGE RETURN (TERMINAL).
C
C 2. SENTINELS -- THE FOLLOWING CHARACTERS IN COLUMN 1 DESIGNATE SPECIAL
C    RECORDS:
C      . CONTROL CARD (IDENTIFIES END-OF-FILE)
C      . REMARK CARD (IGNORED BUT LISTED ON OUTPUT)
C
C 3. DELIMITERS -- ALL OTHER RECORDS CONTAIN DATA FIELDS AND/OR COMMENT
C    FIELDS. ORGANIZED INTO BLOCKS. AND SEPARATED BY THE FOLLOWING DELIM-
C    ITERS:
C      . ENDS DATA FIELD (NEXT DATA FIELD FOLLOWS)
C      .. ENDS LAST DATA FIELD IN BLOCK (COMMENT FIELD FOLLOWS)
C      ... ENDS LAST FIELD IN BLOCK (NEXT BLOCK FOLLOWS)
C
C 4. BLOCKS -- EACH BLOCK IS NORMALLY TERMINATED BY A TRIPLE COMMA (...) OR
C    THE END-OF-RECORD. WHICHEVER COMES FIRST.
C
C 5. SPANNED BLOCKS -- (ACTIVATED FOR CURRENT BLOCK BY CALL TO SPANS) IF A
C    PHYSICAL RECORD ENDS WITH A SINGLE COMMA. THEN THE NEXT FIELD FOR THE
C    CURRENT BLOCK IS TAKEN FROM THE NEXT PHYSICAL RECORD.
C
C 6. INPUT CUEING -- THE CUE MESSAGE (MSCUE) IS PRINTED PRIOR TO READING A
C    BLOCK. PROVIDED THE BLOCK IS THE FIRST BLOCK IN A RECORD. OTHERWISE THE
C    CUE MESSAGE IS IGNORED.
C
C
C MACHINE-DEPENDENT CODE
C -----

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**READS  
002**

```

C      DIMENSION & FORMAT SPECS ASSUME 8 CHARS PER SINGLE PRECISION INTEGER.
C
C
C      EXTERNAL REFERENCES
C      -----
C
C      GETOKM      & GET CHARACTER STRING DATA FIELD FROM BUFFER
C      ERION       & INITIATE I/O & WAIT FOR COMPLETION
C      EAPRNT      & PRINT ASCII IMAGE ON TTY OR LINE PRINTER
C      ERPRNT      & PRINT CHARACTER IMAGE ON TTY OR LINE PRINTER
C      HDLOG       & LOG DIAGNOSTIC MESSAGE
C      HDNOTE      & PRINT/COUNT/LOG 'NOTE' DIAGNOSTIC MESSAGE
C      HDWARN      & PRINT/COUNT/LOG 'WARNING' DIAGNOSTIC MESSAGE
C      HDFATL      & PRINT/COUNT/LOG 'FATAL ERROR' DIAGNOSTIC MESSAGE
C      SYSADD      & ADD DISK SYMBOLIC FILE OR ELT TO SYSIN RUNSTREAM
C      SYSOET      & GET NEXT RECORD FROM SYSIN RUNSTREAM
C      OPEN4       & OPEN COMMAND RECALL FILE (UNIT 4)
C      WRITEN      & WRITE TO COMMAND RECALL FILE (UNIT 4)
C      PUTCHR      & PUT CHARACTER INTO CHARACTER STRING
C      MOVCS       & MOVE CHARACTER STRING
C      INTEGER LENPAD  & LENGTH IN CHARACTERS INCL PAD TO WORD BOUNDARY
C      INTEGER ICE    & INTEGER CHARACTER EQUIVALENT
C
C
C      EXCEPTIONS
C      -----
C
C      1. THE FOLLOWING GENERATE WARNING DIAGNOSTICS:
C
C
C      GLOBAL DECLARATIONS
C      -----
C
C      INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
C      INCLUDE KOMLOG.LIST      & COMMON LOG FILE BUFFER, I/O PKT, POINTERS
C      INCLUDE KOMLUS.LIST      & COMMON POINTERS/FLAGS/BUFFER FOR UNIT 5
C      INCLUDE KOMIO.LIST       & COMMON I/O FUNCTIONS
C
C
C      LOCAL DECLARATIONS
C      -----
C
C      INTEGER NDIAGS      & NUMBER OF QUEUED DIAGNOSTICS TO UNQUEUE
C      INTEGER NO          & SEQUENCE NUMBER OF DIAGNOSTIC BEING UNQUEUED
C      INTEGER MSCUE(11)    & ARGUMENT
C      INTEGER KHTENP(14)   & TEMPORARY CHARACTER STRING
C
C
C      PROCEDURE
C      -----
C
C
C      CALL TRACE
C
C
C      INITIALIZE

```



READS  
003

**N-324**

READS  
004

**N-323**

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

REG-NOM/LSAT-1-ERT  
001

REG-NOM/LSAT-1-ERT . NOMINAL REGISTRATION PARAMETERS FOR LSAT-1, ERT GEOMETRY

SLSTNER  
NERTS (1) = +1  
NERTS (2) = +0  
NERTS (3) = +0

SEND  
SLSTORR  
NERLIN = +2340  
NERSAM = +3240  
ALTKM = .90950150E+03  
ALTSAM = .16020176E+05  
CTRLIN = .11705000E+04  
CTRSAM = .16205000E+04  
CYRLAT = .30263452E+02  
CYRLON = .95308495E+02  
DIRLAT = .00000000E+00  
DIRLON = .00000000E+00  
PITDEG = .00000000E+00  
ROLDEG = .00000000E+00  
YAWDEG = .00000000E+00  
MERGEO = 11129888613

SEND  
SLSTSCN  
SCNA = .38404500E+00  
SCNB = -.25132000E+00  
SCNC = .95505000E-01  
ROLRAC = .95505000E-01  
SCNTH2 = .10074900E+00  
SCNT1W = .16392152E-03

SEND  
SLSTFIT  
NCTLPT = +0  
PCTCTL = .00000000E+00  
RMSMET = .50000000E+04  
UTMCHO = .00000000E+00  
STMCHO = .95309999E+02  
CORSTH (1) = -.78092740E+02  
CORSTH (2) = -.10265850E+02  
CORSTH (3) = .34558722E+07  
CORSTH (4) = -.19468061E+02  
CORSTH (5) = .55854193E+02  
CORSTH (6) = .43244014E+06  
STHCOR (1) = -.12244316E-01  
STHCOR (2) = -.22504722E-02  
STHCOR (3) = .43287984E+05  
STHCOR (4) = -.42673358E-02  
STHCOR (5) = .17119434E-01  
STHCOR (6) = .73442367E+04  
CORSRH (1) = .80325069E+02  
CORSRH (2) = .00000000E+00  
CORSRH (3) = .00000000E+00  
CORSRH (4) = -.50285950E+01

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**REG-NON/LSAT-1-ERT  
002**

**CORSRM (5) = .56789774E+02  
CORSRM (6) = .00000000E+00  
SEND**

DAH PACKAGE APPENDIX N  
UTILITY ROUTINES

REG-NON/LSAT-2-ERT  
001

REG-NON/LSAT-2-ERT . NOMINAL REGISTRATION PARAMETERS FOR LSAT-2. ERT GEOMETRY

SLSTNER  
NERTS (1) = +2  
NERTS (2) = +0  
NERTS (3) = +0

SEND  
SLSTORB  
NERLIN = +2340  
NERSAM = +3264  
ALTKM = .91029568E+03  
ALTSAM = .16131788E+05  
CTRLIN = .11705000E+04  
CTRSAM = .16325000E+04  
CTRLAT = .30263758E+02  
CTRLON = .95304989E+02  
DIRLAT = .00000000E+00  
DIRLON = .00000000E+00  
PITDEG = .00000000E+00  
ROLDEG = .00000000E+00  
YAWDEG = .00000000E+00  
NERGEO = 11129868613

SEND  
SLSTSCN  
SCNA = .38954000E+00  
SCNB = -.26725000E+00  
SCNC = .97588000E-01  
ROLRAC = .97588000E-01  
SCNTH2 = .10079300E+00  
SCNTIH = .16933172E-03

SEND  
SLSTFIT  
NCTLPT = +0  
PCTCTL = .00000000E+00  
RMSMET = .50000000E+04  
UTMCHD = .00000000E+00  
STIMHD = .95299999E+02  
CORSTH (1) = -.78109284E+02  
CORSTH (2) = -.10205053E+02  
CORSTH (3) = .34559388E+07  
CORSTH (4) = -.19418442E+02  
CORSTH (5) = .95515769E+02  
CORSTH (6) = .43150676E+06  
STHCOR (1) = -.12243074E-01  
STHCOR (2) = -.22505538E-02  
STHCOR (3) = .43282444E+05  
STHCOR (4) = -.42824125E-02  
STHCOR (5) = .17225696E-01  
STHCOR (6) = .73867514E+04  
CORSRH (1) = .80332854E+02  
CORSRH (2) = .00000000E+00  
CORSRH (3) = .00000000E+00  
CORSRH (4) = -.49768033E+01

ORIGINAL PAGE 17  
OF PAPER QUARTER

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**REG-NON/LSAT-2-ERT  
002**

**CORSRM (5) = .38445938E+02  
CORSRM (6) = .00000000E+00**

**SEND**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

REG-NOM/LSAT-2-NOM  
001

REG-NOM/LSAT-2-NOM . NOMINAL REGISTRATION PARAMETERS FOR LSAT-2. NOM GEOMETRY

SLSTNER  
NERTS (1) = +2  
NERTS (2) = +0  
NERTS (3) = +0

SEND  
SLSTOR0  
NERLIN = +2903  
NERSAM = +3390  
ALTKM = .55000000E+06  
ALTSAM = .99000000E+07  
CTRLIN = .14920000E+04  
CTRSAM = .17745000E+04  
CTRLAT = .45863116E+02  
CTRLON = .11721024E+03  
DIRLAT = .00000000E+00  
DIRLON = .00000000E+00  
PITDE0 = .00000000E+00  
ROLDE0 = .00000000E+00  
YAWDE0 = .00000000E+00  
NERDE0 = +14298927429

SEND  
SLSTSCN  
SCNA = .10000000E+01  
SCNB = .50000000E-07  
SCNC = .00000000E+00  
ROLRAC = .00000000E+00  
SCNTH2 = .18950000E-03  
SCNTIW = .10000000E-06

SEND  
SLSTFIT  
NCTLPT = +0  
PCTCTL = .00000000E+00  
RMSNET = .50000000E+04  
UTHCHD = .00000000E+00  
STMCHD = .11721000E+03  
CORSTM (1) = -.55496210E+02  
CORSTM (2) = -.12901264E+02  
CORSTM (3) = .51853339E+07  
CORSTM (4) = -.12846599E+02  
CORSTM (5) = .55489490E+02  
CORSTM (6) = .41629879E+06  
STMCOR (1) = -.17098876E-01  
STMCOR (2) = -.39754988E-02  
STMCOR (3) = .90318895E+05  
STMCOR (4) = -.39586538E-02  
STMCOR (5) = .17101047E-01  
STMCOR (6) = .13407797E+05  
CORSRN (1) = .56963685E+02  
CORSRN (2) = .00000000E+00  
CORSRN (3) = .00000000E+00  
CORSRN (4) = .54767132E-01

**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**REG-NON/LSAT-2-NON  
002**

**CONSRM (5) = .56969518E+02  
CONSRM (6) = .00000000E+00**

**SEND**



DAH PACKAGE APPENDIX N  
UTILITY ROUTINES

REG-NOM/LSAT-2-SOM  
001

REG-NOM/LSAT-2-SOM . NOMINAL REGISTRATION PARAMETERS FOR LSAT-2. SOM GEOMETRY

SLSTNER  
NERTS (1) = +2  
NERTS (2) = +0  
NERTS (3) = +0

SEND  
SLSTOR0  
NERLIN = +2003  
NERSAM = +3300  
ALTKM = .99999999E+06  
ALTSAM = .99999999E+07  
CTRLIN = .14920000E+04  
CTRSAM = .17745000E+04  
CTRLAT = .45863116E+02  
CTRLON = .11721024E+03  
DIRLAT = .00000000E+00  
DIRLON = .00000000E+00  
PITDEG = .00000000E+00  
ROLDEG = .00000000E+00  
YAWDEG = .00000000E+00  
NERGEO = +26110087493

SEND  
SLSTSCN  
SCNA = .10000000E+01  
SCNB = .50000000E-07  
SCNC = .00000000E+00  
ROLRAC = .00000000E+00  
SCNTHZ = .18950000E-03  
SCNTIM = .10000000E-06

SEND  
SLSTFIT  
NCTLPT = +0  
PCTCTL = .00000000E+00  
RMSNET = .50000000E+04  
UTHCHD = .00000000E+00  
STMCHD = .11721000E+03  
CORSTH (1) = -.95496210E+02  
CORSTH (2) = -.12901264E+02  
CORSTH (3) = .91853339E+07  
CORSTH (4) = -.12846599E+02  
CORSTH (5) = .98489490E+02  
CORSTH (6) = .41629879E+06  
STNCOR (1) = -.17098976E-01  
STNCOR (2) = -.38754988E-02  
STNCOR (3) = .80318895E+05  
STNCOR (4) = -.39986538E-02  
STNCOR (5) = .17101047E-01  
STNCOR (6) = .13407797E+05  
CORSRH (1) = .98983885E+02  
CORSRH (2) = .00000000E+00  
CORSRH (3) = .00000000E+00  
CORSRH (4) = .94787132E-01

**DAM PACKAGE APPENDIX H  
UTILITY ROUTINES**

**REG-NON/LSAT-2-SOM  
002**

**CORSRM (5) = .50000000E+02  
CORSRM (6) = .00000000E+00**

**SEND**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

REG-NOM/LSAT-2-UTH  
001

REG-NOM/LSAT-2-UTH . NOMINAL REGISTRATION PARAMETERS FOR LSAT-2. UTH GEOMETRY

SLSTNER  
NERTS (1) = +2  
NERTS (2) = +0  
NERTS (3) = +0

SEND  
SLSTORB  
NERLIN = +2903  
NERSAM = +3300  
ALTKM = .55000000E+06  
ALTSAM = .99999999E+07  
CTRLIN = .14020000E+04  
CTRSAM = .17745000E+04  
CTRLAT = .45003117E+02  
CTRLON = .11721024E+03  
DIRLAT = .00000000E+00  
DIRLON = .00000000E+00  
PIYDE0 = .00000000E+00  
ROLDE0 = .00000000E+00  
YAWDE0 = .00000000E+00  
NERQEO = +20341457221

SEND  
SLSTSCN  
SCNA = .10000000E+01  
SCNB = .50000000E-07  
SCNC = .00000000E+00  
ROLRAC = .00000000E+00  
SCNTH2 = .10000000E-03  
SCNT1W = .10000000E-06

SEND  
SLSTFIT  
NCTLPT = +0  
PCTCTL = .00000000E+00  
RMSNET = .50000000E+04  
UTHCND = .00000000E+00  
STHCND = .11700000E+03  
CORSTH (1) = -.55462103E+02  
CORSTH (2) = -.13047320E+02  
CORSTH (3) = .51055754E+07  
CORSTH (4) = -.12992534E+02  
CORSTH (5) = .55455570E+02  
CORSTH (6) = .40027011E+06  
STHCOR (1) = -.17000400E-01  
STHCOR (2) = -.40204930E-02  
STHCOR (3) = .90222041E+05  
STHCOR (4) = -.40036094E-02  
STHCOR (5) = .17000501E-01  
STHCOR (6) = .13020005E+05  
CORSRH (1) = .55003043E+02  
CORSRH (2) = .00000000E+00  
CORSRH (3) = .00000000E+00  
CORSRH (4) = .54049003E-01

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

REG-NON/LSAT-2-UTM  
002

CORSRM (S) = .50000702E+02  
CORSRM (S) = .00000000E+00

SEND

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

REG-NON/LSAT-3-ERT  
001

REG-NON/LSAT-3-ERT . NOMINAL REGISTRATION PARAMETERS FOR LSAT-3. CRT GEOMETRY

SLSTNER  
NERTS (1) = .3  
NERTS (2) = .9  
NERTS (3) = .9

SEND  
SLSTOR8  
MENLIN = .2340  
MERSAN = .3192  
ALTKM = .91814761E+03  
ALTSAN = .19779831E+05  
CTRLIN = .11709000E+04  
CTRSAN = .19989000E+04  
CTRLAT = .31818718E+02  
CTRLON = .91995435E+02  
DIRLAT = .00000000E+00  
DIRLON = .00000000E+00  
PITDEG = .00000000E+00  
ROLDEG = .00000000E+00  
YANDEG = .00000000E+00  
NERDEG = .11129868613

SEND  
SLSTSCN  
SCNA = .36954000E+00  
SCNB = -.26729000E+00  
SCNC = .97588000E-01  
ROLRAC = .97588000E-01  
SCNTH2 = .10079300E+00  
SCNTIH = .17315243E-03

SEND  
SLSTFIT  
NCTLPT = .3  
PCTCTL = .00000000E+00  
RMSNET = .90060000E+04  
UTNCHD = .00000000E+00  
STNCHD = .98000000E+02  
CORSTH (1) = -.77337948E+02  
CORSTH (2) = -.10263842E+02  
CORSTH (3) = .36046810E+07  
CORSTH (4) = -.19917287E+02  
CORSTH (5) = .57307773E+02  
CORSTH (6) = .43214038E+08  
STNCOR (1) = -.12360898E-01  
STNCOR (2) = -.22118049E-02  
STNCOR (3) = .45512088E+05  
STNCOR (4) = -.42959534E-02  
STNCOR (5) = .16680998E-21  
STNCOR (6) = .82770095E+04  
CORSRH (1) = .79838920E+02  
CORSRH (2) = .00000000E+00  
CORSRH (3) = .00000000E+00  
CORSRH (4) = -.59847572E+01

**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**REQ-NON/LSAT-3-ERT  
002**

**CORSRM (5) = .50217048E+02  
CORSRM (6) = .00000000E+00**

**SEND**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

REG-NOM/LSAT-3-NOM  
001

REG-NOM/LSAT-3-NOM . NOMINAL REGISTRATION PARAMETERS FOR LSAT-3. NOM GEOMETRY

SLSTNER  
NERTS (1) = +3  
NERTS (2) = +0  
NERTS (3) = +0

SEND  
SLSTORB  
NERLIN = +2983  
NERSAN = +3390  
ALTKH = .55999999E+08  
ALTSAN = .99999999E+07  
CTRLIN = .14920000E+04  
CTRSAN = .17745000E+04  
CTRLAT = .45863118E+02  
CTRLON = .11721024E+03  
DIRLAT = .00000000E+00  
DIRLON = .00000000E+00  
PITDEG = .00000000E+00  
ROLODEG = .00000000E+00  
YAWDEG = .00000000E+00  
NERGEO = +14298927429

SEND  
SLSTSCN  
SCNA = .10000000E+01  
SCNB = .50000000E-07  
SCNC = .00000000E+00  
ROLRAC = .00000000E+00  
SCNTM2 = .16950000E-03  
SCNTM = .10000000E-06

SEND  
SLSTFIT  
NCTLPT = +0  
PCTCTL = .00000000E+00  
RMSMET = .50000000E+04  
UTHCHD = .00000000E+00  
STMCHD = .11721000E+03  
CORSTM (1) = -.55498210E+02  
CORSTM (2) = -.12901264E+02  
CORSTM (3) = .51853339E+07  
CORSTM (4) = -.12846599E+02  
CORSTM (5) = .55489490E+02  
CORSTM (6) = .41829879E+06  
STNCOR (1) = -.17098976E-01  
STNCOR (2) = -.39754088E-02  
STNCOR (3) = .90318895E+05  
STNCOR (4) = -.39586538E-02  
STNCOR (5) = .17101047E-01  
STNCOR (6) = .13407787E+05  
CORSRN (1) = .58983885E+02  
CORSRN (2) = .00000000E+00  
CORSRN (3) = .00000000E+00  
CORSRN (4) = .54767132E-01

**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**REC-NON/LSAT-3-MON  
002**

**CONSRM (5) = .56969518E+02  
CONSRM (6) = .00000000E+00**

**SEND**



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

REG-NOM/LSAT-3-SOM  
001

REG-NOM/LSAT-3-SOM . NOMINAL REGISTRATION PARAMETERS FOR LSAT-3. SOM GEOMETRY

SLSTNER

NERTS (1) = +3  
NERTS (2) = +0  
NERTS (3) = +0

SEND

SLSTOR0

NERLIN = +2983  
NERSAM = +3390  
ALTKM = .55889999E+08  
ALTSAM = .99999999E+07  
CTRLIN = .14920000E+04  
CTRSAM = .17745000E+04  
CTRLAT = .45863116E+02  
CYRLON = .11721024E+03  
DIRLAT = .00000000E+00  
DIRLON = .00000000E+00  
PITDE0 = .00000000E+00  
ROLDE0 = .00000000E+00  
YAWDE0 = .00000000E+00  
NEROE0 = +26110087493

SEND

SLSTSCN

SCNA = .10000000E+01  
SCNB = .50000000E-07  
SCNC = .00000000E+00  
ROLRAC = .00000000E+00  
SCNTH2 = .18950000E-03  
SCNT1H = .10000000E-06

SEND

SLSTFIT

NCTLPT = +0  
PCTCTL = .00000000E+00  
RMSMET = .50000000E+04  
UTMCHO = .00000000E+00  
STNCHO = .11721000E+03  
CORSTM (1) = -.55496210E+02  
CORSTM (2) = -.12901264E+02  
CORSTM (3) = .51853339E+07  
CORSTM (4) = -.12846599E+02  
CORSTM (5) = .55488490E+02  
CORSTM (6) = .41829879E+06  
STNCOR (1) = -.17098976E-01  
STNCOR (2) = -.39754988E-02  
STNCOR (3) = .90318895E+05  
STNCOR (4) = -.39586538E-02  
STNCOR (5) = .17101047E-01  
STNCOR (6) = .13407797E+05  
CORSRM (1) = .58963685E+02  
CORSRM (2) = .00000000E+00  
CORSRM (3) = .00000000E+00  
CORSRM (4) = .54767132E-01

**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**REG-NOM/LSAT-3-SOM  
002**

**CORSRM (5) = .56969510E+02  
CORSRM (6) = .00000000E+00**

**SEND**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

REG-NOM/LSAT-3-UTM  
001

REG-NOM/LSAT-3-UTM . NOMINAL REGISTRATION PARAMETERS FOR LSAT-3. UTM GEOMETRY

SLSTNER  
NERTS (1) = +3  
NERTS (2) = +0  
NERTS (3) = +0

SEND  
SLSTOR0  
NERLIN = +2903  
NERSAM = +3390  
ALTKM = .55999999E+06  
ALTSAM = .99999999E+07  
CTRLIN = .14920000E+04  
CTRSAM = .17745000E+04  
CTRLAT = .45863117E+02  
CTRLON = .11721024E+03  
DIRLAT = .00000000E+00  
DIRLON = .00000000E+00  
PITDEG = .00000000E+00  
ROLDEG = .00000000E+00  
YAMDEG = .00000000E+00  
NERGEO = +28341457221

SEND  
SLSTSCN  
SCNA = .10000000E+01  
SCNB = .50000000E-07  
SCNC = .00000000E+00  
ROLRAC = .00000000E+00  
SCNTH2 = .16950000E-03  
SCNTIM = .10000000E-06

SEND  
SLSTFIT  
NCTLPT = +0  
PCTCTL = .00000000E+00  
RMSNET = .90000000E+04  
UTMCMD = .00000000E+00  
STMCMD = .11700000E+03  
CORSTH (1) = -.55462103E+02  
CORSTH (2) = -.13047320E+02  
CORSTH (3) = .51855754E+07  
CORSTH (4) = -.12992534E+02  
CORSTH (5) = .55455578E+02  
CORSTH (6) = .40027811E+06  
STHCOR (1) = -.17088466E-01  
STHCOR (2) = -.40204938E-02  
STHCOR (3) = .90222841E+05  
STHCOR (4) = -.48036084E-02  
STHCOR (5) = .17090501E-01  
STHCOR (6) = .13920085E+05  
CORSRM (1) = .56963843E+02  
CORSRM (2) = .00000000E+00  
CORSRM (3) = .00000000E+00  
CORSRM (4) = .54849863E-01

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

REG-NON/LSAT-3-UTM  
002

CORSRN (5) = .58989762E+02  
CORSRN (6) = .00000000E+00

SEND

REVERT  
OO1

```

C -----
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      12/20/72      ORIGINAL CODE
C      E H SCHLOSSER      LEC      08/08/78      UPGRADED DOCUMENTATION
C
C METHOD
C -----
C
C      REVERT EQUATION 1 TO EQUATION 2:
C
C      BX=A(1)*AX+A(2)*AY+A(3)      BY=A(4)*AX+A(5)*AY+A(6)      (1)
C
C      AX=B(1)*BX+B(2)*BY+B(3)      AY=B(4)*BX+B(5)*BY+B(6)      (2)
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      NONE.
C
C EXCEPTIONS
C -----
C
C      1. NO CHECK IS MADE FOR DIVISION BY ZERO.
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C LOCAL DECLARATIONS
C -----
C
C      DIMENSION A(6),B(6)      2 ARGUMENTS
C      REAL FAC      2
C
C PROCEDURE
C -----

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

REVERT  
002

C  
C

```
CALL TRACE
FAC=1.0/(A(1)*A(5)-A(2)*A(4))
B(1)=+FAC*A(5)
B(2)=-FAC*A(2)
B(3)=+FAC*(A(2)*A(6)-A(3)*A(5))
B(4)=-FAC*A(4)
B(5)=+FAC*A(1)
B(6)=-FAC*(A(1)*A(6)-A(3)*A(4))
RETURN
END
```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

RITADD  
001

```

SUBROUTINE RITADD( 8 WRITE OR ADD SYMBOLIC ELEMENT
1 IUNIT,          8 OUTPUT UNIT NUMBER (0 MEANS ADD)
1 NAMELT,         8 COMPLETE ELEMENT NAME
1 KEY,            8 RETRIEVAL KEY(S) AND POINTERS
8 $)              8 ERROR TRANSFER LABEL

```

```

C -----
C
C (E H SCHLOSSER)
C
C
C ANY ONE OF THE FOLLOWING CONDITIONS CAUSES AN ERROR RETURN:
C 1. FILE/ELEMENT/VERSION NAME CONTAINS EXCESS CHARACTERS
C 2. FILE NOT SPECIFIED
C 3. FILE DOES NOT EXIST
C 4. ELEMENT IN CATALOGED FILE DOES NOT EXIST
C
C ANY ONE OF THE FOLLOWING CONDITIONS ABORTS THE RUN:
C 1. ELEMENT IN TEMPORARY FILE DOES NOT EXIST
C 2. NESTED $ADD ELEMENT DOES NOT EXIST
C
C THE ELEMENT MAY NOT CONTAIN ANY CONTROL STATEMENTS OTHER THAN $ADD.
C
C
C EXTERNAL SUBROUTINES/FUNCTIONS CALLED
C -----
C
C MOFATL
C ERCSF
C ERREAD
C EAPRNT
C
C
C INCLUDE KOMXQT.LIST
C INCLUDE ASHDEF.LIST
C DIMENSION NAMELT(1),KEY(1),KDELIM(3),IMAGE(22)
C DATA KDELIM/'00000','00000','00000/' 8 QUAL=FILE.ELT/VER
C DEFINE IMAGE(N)=IMAGE(N)
C CALL TRACE
C
C
C MOVE COMPLETE ELEMENT NAME TO IMAGE
C
C IMAGE(2)='
C DO 130 NWD=1,10
C IMAGE(NWD+2)=NAMELT(NWD)
C IMAGE(NWD+3)=' 8 CSFS STOP CHARACTER
C IF((NAMELT(NWD).EQ.-0).OR. 8 END OF HOLLERITH LITERAL
C & (NAMELT(NWD).EQ.' 1) GO TO 200
C 130 CONTINUE
C
C
C CHECK QUALIFIER/FILE/ELEMENT/VERSION NAMES FOR ILLEGAL/EXCESS CHARACTERS
C
C 200 NPART=0
C NCHAR=0
C DO 250 NWD=3,12

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

R1TADD  
002

```

DO 240 NBIT=0.30.0
  IF((FLO(NBIT,0,IMAGE(NWD)).EQ.'33330 ').AND.
    & (NCHAR.EQ.0)) GO TO 240      & IGNORE LEADING BLANKS
  IF((FLO(NBIT,0,IMAGE(NWD)).GE.'88888A').AND.
    & (FLO(NBIT,0,IMAGE(NWD)).LE.'88888Z')) GO TO 230
  IF((FLO(NBIT,0,IMAGE(NWD)).GE.'888880').AND.
    & (FLO(NBIT,0,IMAGE(NWD)).LE.'888889')) GO TO 230
  IF((FLO(NBIT,0,IMAGE(NWD)).EQ.'88888-').OR.
    & (FLO(NBIT,0,IMAGE(NWD)).EQ.'888885')) GO TO 230
  NCHAR=0
210 NPART=NPART+1
  IF(NPART.GT.3) GO TO 220
  IF(FLO(NBIT,0,IMAGE(NWD)).EQ.KDELIM(NPART)) GO TO 240
  GO TO 210
220 FLO(NBIT,0,IMAGE(NWD))='88888 '      & TERMINATE NAME
  GO TO 400
230 NCHAR=NCHAR+1
  IF(NCHAR.GT.12) GO TO 220
240 CONTINUE
250 CONTINUE
  GO TO 000

C
C
C CHECK IF ELEMENT EXISTS
C
400 IMAGE(1)='8START'
  CALL ERCSF(LSTAT,IMAGE)
  LSTAT=FLO(33.3,LSTAT)
  GO TO(500,620,630,640,630,500).LSTAT

C
C
C ADD ELEMENT
C
500 IF(IUNIT.NE.0) GO TO 600
  IMAGE(1)='8ADD '
  CALL ERCSF(NA0,IMAGE)
  GO TO 600

C
C
C READ LINE
C
600 IMAGE(1)='8ADD.E'
  CALL ERCSF(NA0,IMAGE)
  NBLNK=0
  NPRINT=0
620 CALL ERREAD(9900,IMAGE,NWDH1)
  NWDH1=ASMMH2(NWDH1)
  IMAGE(NWDH1+1)=' '

C
C
C CHECK IF SPECIFIED WORDS OF LINE MATCH KEYS
C
DO 840 NWD=1.11.2
  IF((KEY(NWD).LT.1).OR.(KEY(NWD).GT.22)) GO TO 650
  IF(AND(IMAGE(KEY(NWD))-'8',KEY(NWD+1)-'8').NE.(KEY(NWD+1)-'8'))
    & GO TO 620

```



DAN PACKAGE APPENDIX H  
UTILITY ROUTINES

RITADD  
003

```

      040 CONTINUE
C
C
C SPECIAL PROCESSING FOR DEMAND MODE TO UNIT 6 ONLY
C
      050 IF(IMAGE.NE.0) GO TO 070
      IF(IUNIT.NE.6) GO TO 070
      IF(ASHSI(IMAGE(1)).EQ.'00000.') GO TO 020      & DO NOT PRINT
      IF(NWDH.NE.0) GO TO 064      & NOT A BLANK LINE
      NBLNK=NBLNK+1
      GO TO(020,062,070,085).NBLNK
      062 IF(NPRNT.LT.14) GO TO 070
      063 CALL EAPRNT(1,15,      & ASCII: 15*(NUL.NUL.NUL.NUL)
      &
      &
      NPRNT=NPRNT-32
      IF(NPRNT.GT.14) GO TO 063
      NPRNT=0
      GO TO 020
      064 NPRNT=NPRNT+NWD
      065 NBLNK=0
C
C
C WRITE LINE
C
      070 WRITE(IUNIT,075) (IMAGE(NWD),NWD=1,NWDH(1)
      075 FORMAT(22A6)
      GO TO 020
C
C
C ERROR RETURN
C
      020 CALL HDFATL('RITADD FILE NOT CATALOGED')
      GO TO 080
      030 GO TO 080      & RITADD ELEMENT DOES NOT EXIST
      040 CALL HDFATL('RITADD FILE NOT SPECIFIED')
      080 RETURN &
C
C
C NORMAL RETURN
C
      090 RETURN
      END

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**RL2ISX  
001**

```

      SUBROUTINE RL2ISX( 0 CONVERT REAL TO SEXAGENARY
      I RL,              0 REAL NUMBER
      0 ISX,             0 INTEGER ARRAY CONTAINING SEX'Y NUMBER (I PLACE/WORD)
      I NSP,             0 NUMBER OF SEX'Y PLACES IN NUMBER (NSP>1)
      0 REMAIN)          0 REAL REMAINDER
      -----
C
C
C (E H SCHLOSSER)
C
      DIMENSION ISX(1)
      REMAIN=ABS(RL)
      DO 200 N=1,NSP
      ISX(N)=IFIX(REMAIN)
      REMAIN=(REMAIN-ISX(N))*60.
200 CONTINUE
      REMAIN=REMAIN/60.
      ISX(1)=ISIGN(ISX(1),RL)
      RETURN
      END

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

RLRSX  
201

```
      SUBROUTINE RLRSX( 8 CONVERT REAL TO SEXAGENARY
1  RL,                8 REAL NUMBER
8 SX,                8 REAL ARRAY CONTAINING SEX'Y NUMBER (1 PLACE/WORD)
1 NSP)              8 NUMBER OF SEX'Y PLACES IN NUMBER (NSP>1)
-----
C
C
C (E H SCHLOSSER)
C
      DIMENSION SX(1)
      REMAIN=ABS(RL)
      DO 200 N=2,NSP
      SX(N-1)=AINT(REMAIN)
      REMAIN=(REMAIN-SX(N-1)*60.
200 CONTINUE
      SX(NSP)=REMAIN
      SX(1)=SIGN(SX(1),RL)
      RETURN
      END
```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

RL4SX  
001

FUNCTION RL4SX(    : REAL FUNCTION FOR SEXAGENARY ARGUMENT  
: SX.               : REAL ARRAY CONTAINING SEXAGENARY NUMBER (1 PLACE/WORD)  
: NSP.              : NUMBER OF SEXAGENARY PLACES IN NUMBER (NSP>0)  
: S)                 : ERROR TRANSFER LABEL

C  
C  
C (E H SCHLOSSER)  
C

```

      DIMENSION SX(1)
      DEFINE RECBAS=1./60.
      RL4SX=ABS(SX(1))
      PWRBAS=RECBAS
      N=1
      GO TO 200
100  N=N+1
      IF(SX(N).GE.60.) GO TO 800
      RL4SX=RL4SX+SX(N)*PWRBAS
      PWRBAS=PWRBAS*RECBAS
200  IF(N.LT.NSP) GO TO 100
      RL4SX=SIGN(RL4SX,SX(1))
      RETURN
800  RETURN 3
      END

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

ROTCHX  
001

```

SUBROUTINE ROTCHX(      & ROTATE PAIR OF COLUMNS TO MAXIMIZE FUNCTION
I AMAT.                & ORIGINAL SINGLE PRECISION MATRIX
I FUNCHX.              & NAME OF EXTERNAL FUNCTION TO MAXIMIZE
I NRUSE,NCUSE.         & NUMBER OF ROWS & COLUMNS USED
I NRDIM,NCDIM.         & NUMBER OF ROWS & COLUMNS DIMENSIONED
I NC1ROT,NC2ROT.       & COLUMN NUMBERS OF COLUMNS TO ROTATE
O THETA.               & ROTATION ANGLE IN RADIANS WHICH MAXIMIZES FUNCTION
O RHAT)               & ROTATED MATRIX (MUST NOT BE SAME AS AMAT)
-----
C
C
C (M L BROWN)
C
C
C FUNCHX ARGUMENTS:
C (RHAT,NRUSE,NCUSE,NRDIM,NCDIM,NC1ROT,NC2ROT)
C
C
C      DIMENSION AMAT(NRDIM,NCDIM),RHAT(NRDIM,NCDIM)
C      DIMENSION CUMROT(3),FMAX(3)
C      CALL TRACE
C
C
C ASSIGN INITIAL VALUES OF ROTATION ANGLE IN DESIRED RANGE
C
C      CUMROT(1)=0.00      & 0 DEG+-
C      CUMROT(2)=+.45     & 26 DEG+-
C      CUMROT(3)=+.90     & 52 DEG+-
C
C
C
C COMPUTE INITIAL VALUES OF FUNCTION TO MAXIMIZE
C
C      DO 200 MODIT=1,3
C      CALL ROTCOL(AMAT,NRUSE,NCUSE,NRDIM,NCDIM,NC1ROT,NC2ROT,
C      & CUMROT(MODIT),RHAT)
C      FMAX(MODIT)=FUNCHX(RHAT,NRUSE,NCUSE,NRDIM,NCDIM,NC1ROT,NC2ROT)
C 200 CONTINUE
C
C
C INSURE THAT INITIAL ROTATION ANGLES ARE NEAR FUNCTION MAXIMUM
C
C      IF(FMAX(1).LT.FMAX(2)) GO TO 250      & CUMROT(1) IS TO LEFT OF MAXIMUM
C      CUMROT(3)=+.8      & SHIFT INITIAL ROTATION ANGLES .4 RADIANS TO LEFT
C      CALL ROTCOL(AMAT,NRUSE,NCUSE,NRDIM,NCDIM,NC1ROT,NC2ROT,
C      & CUMROT(3),RHAT)
C      FMAX(3)=FUNCHX(RHAT,NRUSE,NCUSE,NRDIM,NCDIM,NC1ROT,NC2ROT)
C      GO TO 300
C
C 250 IF(FMAX(3).LT.FMAX(2)) GO TO 300      & CUMROT(3) IS TO RIGHT OF MAXIMUM
C      CUMROT(1)=+.8      & SHIFT INITIAL ROTATION ANGLES .4 RADIANS TO RIGHT
C      CALL ROTCOL(AMAT,NRUSE,NCUSE,NRDIM,NCDIM,NC1ROT,NC2ROT,
C      & CUMROT(1),RHAT)
C      FMAX(1)=FUNCHX(RHAT,NRUSE,NCUSE,NRDIM,NCDIM,NC1ROT,NC2ROT)
C
C
C ITERATE TO CONVERGE TO ROTATION ANGLE WHICH YIELDS FUNCTION EXTREMUM
C

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

ROTCHX  
002

```

300 DO 400 NITER=0.50
    MITOLD=MOD(NITER,3)+1
    MITNEW=MOD(NITER+1,3)+1
C
C
C FIT QUADRATIC & SOLVE FOR EXTREMUM TO ESTIMATE NEXT ROTATION ANGLE
C
    AVGROT=(CUMROT(1)+CUMROT(2)+CUMROT(3))/3.
    CALL QUAD(CUMROT,FMAX,A,B,C,CUMROT(MITNEW))
    CUMROT(MITNEW)=AMINO(CUMROT(MITNEW),AVGROT+.2)      & LIMIT OVERSHOOT!
    CUMROT(MITNEW)=AMAXO(CUMROT(MITNEW),AVGROT-.2)      & LIMIT OVERSHOOT!
C
C ROTATE & COMPUTE FUNCTION TO MAXIMIZE
C
    CALL ROTCOL(AMAT,NRUSE,NCUSE,NRODM,NCDIM,NC1ROT,NC2ROT,
    & CUMROT(MITNEW),RMAT)
    FMAX(MITNEW)=FUNCMX(RMAT,NRUSE,NCUSE,NRODM,NCDIM,NC1ROT,NC2ROT)
C
C
C CHECK FUNCTION FOR CONVERGENCE
C
    IF(ABS(FMAX(MITNEW)-FMAX(MITOLD)).LE.ABS(FMAX(MITNEW))*1.E-6)
    & GO TO 900
400 CONTINUE
C
C
C FLAG NO CONVERGENCE
C
    CALL MDHARN('NO CONVERGENCE IN ROTCHX')
C
C
C RETURN WITH FINAL ROTATION ANGLE
C
900 THETA=CUMROT(MITNEW)
    RETURN
    END

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

ROTCOL  
001

```

      SUBROUTINE ROTCOL( 3 ROTATE PAIR OF COLUMNS IN MATRIX
      1 BMAT.           3 ORIGINAL SINGLE PRECISION MATRIX
      1 NRUSE,NCUSE.     3 NUMBER OF ROWS & COLUMNS USED
      1 NRDIM,NCDIM.     3 NUMBER OF ROWS & COLUMNS DIMENSIONED
      1 NC1ROT,NC2ROT.   3 COLUMN NUMBERS OF COLUMNS TO ROTATE
      1 THETA.           3 ROTATION ANGLE IN RADIANS
      0 CHAT)           3 NEW MATRIX (MUST NOT BE SAME AS BMAT)
-----
C
C
C (E H SCHLOSSER)
C
C
      DIMENSION BMAT(NRDIM,NCDIM),CHAT(NRDIM,NCDIM)
      DIMENSION ROT(2,2)
      CALL TRACE(KHR4IN(IFIX(THETA*1000.)))      3 MILLI-RADIANS
C
C
C COPY UNROTATED COLUMNS FROM BMAT TO CHAT
C
      DO 240 NC=1,NCUSE
      IF(NC.EQ.NC1ROT) GO TO 240
      IF(NC.EQ.NC2ROT) GO TO 240
      DO 220 NR=1,NRUSE
      220 CHAT(NR,NC)=BMAT(NR,NC)
      240 CONTINUE
C
C
C FILL ROTATION MATRIX
C
      ROT(1,1)=COS(THETA)
      ROT(1,2)=SIN(THETA)
      ROT(2,1)=-SIN(THETA)
      ROT(2,2)=COS(THETA)
C
C
C PERFORM ROTATION
C
      NCROT=NC1ROT
      DO 460 NC=1,2
      DO 440 NR=1,NRUSE
      CHAT(NR,NCROT)=0.
      DO 420 NIP=1,2
      420 CHAT(NR,NCROT)=CHAT(NR,NCROT)+BMAT(NR,NIP)*ROT(NIP,NC)
      440 CONTINUE
      460 NCROT=NC2ROT
C
C
      RETURN
      END

```

ROTRON  
001

**N-355**



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

R3TREC  
001

```

SUBROUTINE R3TREC(      8 READ ONE IMAGE RECORD
O ITAPBF.              8 MSS/RBY BUFFER
I NWIBF.              8 NUMBER OF WORDS IN 1 BUFFER
O ITSTAT.             8 I/O STATUS CODES
                        ' ' NORMAL COMPLETION
                        'EOF' END OF FILE
                        'BADR' BAD RECORD
                        'BADF' BAD FILE

I MVDIR.              8 DIRECTION OF READ
I MINTRV,MAXTRV.      8 MIN. MAX # BYTES IN A TRIVIAL RECORD
I MAXSMT)             8 MAX # BYTES IN A SHORT RECORD
-----

C
C
C
C HISTORY
C -----
C   J C CRISP          LEC    08/29/79    REQUIREMENTS
C   MARY TOMPKINS      LEC    08/30/79    ALGORITHM DESIGN
C   MARY TOMPKINS      LEC    08/30/79    ALGORITHM CODING
C
C
C METHOD
C -----
C
C   SET MX3BLK TO 5. IOADDR(LU3PKT) = LOCATION OF BUFFER AND
C   IOSIZE(LU3PKT) = NWIBF. CHECK MVDIR FOR DIRECTION OF READ.
C   READ RECORD. SET ITSTAT = ICODE(LU3PKT). IF ITSTAT = 'EOF'
C   OR 'BADF', RETURN. CHECK FOR TRIVIAL RECORD. A RECORD IS
C   CONSIDERED TRIVIAL IF ITS LENGTH IS BETWEEN MINTRV AND MAXTRV
C   INCLUSIVE. IF TRIVIAL RECORD IGNORE IT AND READ AGAIN. CHECK
C   FOR SHORT BLOCK. IF SHORT BLOCK, ISSUE QUEUED NOTE AND READ
C   AGAIN. IF NUMBER OF SHORT BLOCKS > MXSBLK, ISSUE NOTE. SET
C   ITSTAT = 'BADF', AND RETURN. ELSE, IF LU3BFH = '88', CALL
C   BST488.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C   NONE
C
C
C EXTERNAL REFERENCES
C -----
C
C   ERION      8 INITIATE I/O AND WAIT FOR COMPLETION
C   MNOTE      8 PRINT/LOG/COUNT 'NOTE' MESSAGES
C   BST488     8 '88' TO 'BST' (INTERNAL) FORMAT
C   INTEGER NBYN1 8 NO. BYTES FOR NO. INTEGERS
C   DOUBLE PRECISION COS4IN 8 VARIABLE LENGTH CHAR STRING FOR INTEGER
C
C
C EXCEPTIONS
C -----
C

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

R3TREC  
002

```

C      1. THE FOLLOWING CONDITIONS GENERATE THE OUTPUT SHOWN:
C          SHORT BLOCK      NOTE/LENGTH/LINE NO.
C          > THAN MAX # OF SHORT BLOCKS      NOTE/LINE NO.
C      2. IF THE NUMBER OF WORDS READ IS LESS THAN MAXSHT OR NOT
C          EQUAL TO NHIBF, THIS ROUTINE WILL IGNORE THE READ AS A SHORT
C          BLOCK.
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOHXQT.LIST      % COMMON PROGRAM EXECUTION SWITCHES.COUNTERS
C      INCLUDE KOHIO.LIST      % FORTRAN MANIPULATION OF ASSEMBLER I/O PKT.
C      INCLUDE PXBDEF.LIST      % DEFINITION OF INTERNAL BUFFER STRUCTURE
C      INCLUDE KOHLU3.LIST      % I/O AND UNPACKING DATA FOR MSS/RBV DATA
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER ITAPBF(NHIBF)      % MSS/RBV BUFFER
C      INTEGER MXSBLK              % MAX NO. OF SHORT BLOCKS ALLOWED
C
C PROCEDURE
C -----
C
C SET POINTERS IN LU3PKT TO ALLOW READ OF ONE RECORD
C
C      MXSBLK = 5
C      IOADDR(LU3PKT) = LOC(ITAPBF)
C      IOSIZE(LU3PKT) = NHIBF
C
C READ: IGNORE UP TO MXSBLK # OF SHORT BLOCKS
C
C      NBLKS=MXSBLK+1
C      DO 200 MV = 1,NBLKS
150      IF(MVDIR.EQ.1)IOFUNC(LU3PKT) = 'BK' % READ FORWARD
C          IF(MVDIR.EQ.-1)IOFUNC(LU3PKT) = 'BL' % READ BACKWARD
C          CALL ERION(LU3PKT)
C          ITSTAT = IOCODE(LU3PKT)
C          IF( (ITSTAT.EQ.'EOF').OR.(ITSTAT.EQ.'BADF') ) GO TO 900
C          NBYTES=NB4NI(IONWDS(LU3PKT))
C          IF(LU3PFH.EQ.'88') NBYTES=NBYTES*9/8
C          IF (NBYTES.GE.MINTRV .AND. NBYTES.LE.MAXTRV) GO TO 150 % TRIVIAL REC
C          IF (NBYTES.GT.MAXSHT .OR. NBYTES.GE.NB4NI(NHIBF)) GO TO 500
C          CALL MONOTE('SHORT BLOCK'.C834IN(NBYTES,5),
C              'BYTES AT LINE'.C834IN(LU3LBF,5),'')
200 CONTINUE
C
C IF MAX # SHORT BLOCKS EXCEEDED SET STATUS TO 'BADF'
C
C      CALL MONOTE('MAXIMUM NUMBER OF SHORT BLOCKS EXCEEDED AT LINE #',

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

ASTREC  
003

```
      COS4(IN(LU3LBF.9).')' )  
      ITSTAT = 'BADF'  
      GO TO 900
```

C

C

C CHECK TO DETERMINE IF FORMAT IS IN INTERNAL FORMAT

C

900 CONTINUE

IF(LU3BFH.EQ.'BB')CALL B5T4B8(ITAPBF, ITAPBF.NB4N1(NH1BF))

C

C

900 RETURN

C

END

C

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

SETHOD  
001

```

SUBROUTINE SETHOD(  & GET/SET MODE SWITCHES
  &
  I NCOND)  & 0 (OFF) OR 1 (ON)
  -----
C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      08/27/79      ORIGINAL CODE
C      E M SCHLOSSER      LEC      08/20/78      ADD LEGEND MODE SWITCH
C      E M SCHLOSSER      LEC      02/23/79      ADD NULCST & DELETE REF TO CKFLDS
C      E M SCHLOSSER      LEC      12/08/79      ADD COLOR MODE & STREAM I/O
C      E M SCHLOSSER      LEMSCO   06/25/80      ADD LOCAL MODE
C
C
C
C METHOD
C -----
C
C      GET/CHECK USER MODE SPECIFICATION & SET ITS SWITCH TO NCOND.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      GETSKM      GET USER MODE SPECIFICATION
C      WARNS      GENERATE WARNING DIAGNOSTIC FOR UNIT 5 INPUT
C      CBINIT      & INITIALIZE CHARACTER BUFFER
C      CB4CST      & CHARACTER BUFFER FOR CHARACTER STRING
C      INTEGER NI4NC  & NUMBER OF INTEGERS FOR NUMBER OF CHARACTERS
C
C
C EXCEPTIONS
C -----
C
C      1. AN INVALID MODE SPECIFICATION GENERATES A DIAGNOSTIC.
C
C      2. ANY NUMBER OF MODE SPECIFICATIONS (INCLUDING NONE) ARE ALLOWED.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST      & COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
C      INCLUDE NULCST.LIST      & DEFINE NULL CHARACTER STRING
C      INCLUDE ICBUF1      & DECLARE CHARACTER BUFFER
C
C
C LOCAL DECLARATIONS
C -----

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

SETMOD  
002

```

C      INTEGER INTMP      & TEMPORARY
C      INTEGER KHTMP      & TEMPORARY
C
C      C PROCEDURE
C      C -----
C
C      CALL TRACE
C
C      C GET/CHECK MODE NAME AND SET SWITCH TO NCOND
C
C      100 CONTINUE
C          KHTMP=' NUL '
C          CALL GETSKH(KHTMP,3,NULCST)
C          IF(KHTMP.EQ.' NUL ') GO TO 800      & NO SPECIFICATION
C              IF(KHTMP.NE.'BAT') GO TO 220      & BATCH
C                  MBATCH=NCOND
C                  GO TO 100
C      220      IF(KHTMP.NE.'CHE') GO TO 240      & CHECKOUT
C                  MCHECK=OR(NCOND,MDATA)
C                  (TURN OFF CHECKOUT ONLY IF NOT DATA/CHECKOUT)
C                  NDTOTL=NDWARN+NDFATL+MCHECK
C                  (PREVENTS PROCESSING WINDOW IN CHECKOUT MODE)
C                  GO TO 100
C      240      IF(KHTMP.NE.'COL') GO TO 260      & COLOR
C                  MCOLOR=NCOND
C                  GO TO 100
C      260      IF(KHTMP.NE.'CON') GO TO 280      & CONFIRM
C                  MCFIRM=NCOND
C                  GO TO 100
C      280      IF(KHTMP.NE.'DUM') GO TO 300      & DUMP
C                  MDUMP=NCOND
C                  GO TO 100
C      300      IF(KHTMP.NE.'ECH') GO TO 320      & ECHO
C                  MECHO=NCOND
C                  GO TO 100
C      320      IF(KHTMP.NE.'LEO') GO TO 330      & LEGEND
C                  MLEOND=NCOND
C                  GO TO 100
C      330      IF(KHTMP.NE.'LOC') GO TO 340      & LOCAL
C                  IF(MLOCAL.EQ.NCOND) GO TO 100      & NO CHANGE
C                      IF(NCOND.NE.0) GO TO 335
C                          INTMP=MADLEV
C                          MSHTCH(1)=MSHSAV(1)
C                          MSHTCH(2)=MSHSAV(2)
C                          MADLEV=INTMP
C                          MLOCAL=0
C                          GO TO 100
C      335      CONTINUE      & TURN ON
C                          MSHSAV(1)=MSHTCH(1)
C                          MSHSAV(2)=MSHTCH(2)
C                          MLOCAL=1
C                          GO TO 100
C      340      IF(KHTMP.NE.'PRO') GO TO 360      & PROMPT

```

DAN PACKAGE APPENDIX H  
UTILITY ROUTINES

SETMOD  
003

```

                                NPROMT=NCOND
                                GO TO 100
300      IF(KHTEMP.NE.'TRA') GO TO 300      & TRACE
                                NTRACE=NCOND
                                GO TO 100
300      CALL WARNB('BAD MODE --')
                                GO TO 100
C
C
C CONFIRM MODE SETTINGS
C
000 CALL CBINIT(ICBUF1)
    IF(NCOND.EQ.0) CALL CB4CST(ICBUF1, ' OFF')
    IF(NCOND.NE.0) CALL CB4CST(ICBUF1, ' ON')
    IF(MBATCH.EQ.NCOND) CALL CB4CST(ICBUF1, ' BATCH')
    IF(MCHECK.EQ.NCOND) CALL CB4CST(ICBUF1, ' CHECKOUT')
    IF(MCOLOR.EQ.NCOND) CALL CB4CST(ICBUF1, ' COLOR')
    IF(MCFIRM.EQ.NCOND) CALL CB4CST(ICBUF1, ' CONFIRM')
    IF(MDUMP.EQ.NCOND) CALL CB4CST(ICBUF1, ' DUMP')
    IF(MECHO.EQ.NCOND) CALL CB4CST(ICBUF1, ' ECHO')
    IF(MLEND.EQ.NCOND) CALL CB4CST(ICBUF1, ' LEGEND')
    IF(MLOCAL.EQ.NCOND) CALL CB4CST(ICBUF1, ' LOCAL')
    IF(MPROMT.EQ.NCOND) CALL CB4CST(ICBUF1, ' PROMPT')
    IF(MIN0(TRACE.1).EQ.NCOND) CALL CB4CST(ICBUF1, ' TRACE')
    IF(MCFIRM.NE.0) CALL ERPRNT(1,N14NC(120),ICBUF1)      & PRINT THE BUFFER
C
C
C NORMAL RETURN
C
    RETURN
    END

```

**SHABAN**  
**001**

.....

C	C A HELMKE	LEC	09/19/79	REQUIREMENTS
C	C A HELMKE	LEC	10/01/79	ALGORITHM DESIGN
C	C A HELMKE	LEC	10/19/79	ALGORITHM CODING

```

C
C      COPY MPXRAM BUFFER TO MPXSHA BUFFER.  SEARCH FOR EXTREMA PIXELS
C      IN MPXRAM FROM MPXRAM(PXLBIN+2) TO MPXRAM(PXHBIN-2).  AT EXTREMA
C      PIXELS COMPUTE A SHARPENED VALUE AND STORE IN CORRESPONDING PIXEL
C      OF MPXSHA.

```

C  
C  
C      NONE.

```

C      EXTERNAL  GETBYT      & GET NON-NEGATIVE INTEGER FROM BYTE STRING
      EXTERNAL  PUTBYT      & PUT ONE NON-NEGATIVE INTEGER INTO BYTE STRING
      EXTERNAL  GETICE      & GET INTEGER-CHAR-EQUIVALENT FROM CHAR STRING
      EXTERNAL  PUTICE      & PUT INTEGER-CHAR-EQUIVALENT INTO CHAR STRING
      EXTERNAL  GETINT      & GET INTEGER FROM INTEGER STRING
      EXTERNAL  PUTINT      & PUT INTEGER INTO INTEGER STRING
      DOUBLE PRECISION COS4CS
C      COS4CS      & VARIABLE CHARACTER STRING FOR FIXED CHARACTER STRING

```

1. IF THIS SUBROUTINE IS CALLED WITH THE SAME BUFFER FOR MPXSHA  
AND MPXRAN THE RESULTS IN MPXSHA ARE INCORRECT.

## 2

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**SHASAM  
002**

```

INCLUDE KONKBT.LIST      8 COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
INCLUDE PXBDEF.LIST      8 DEFINE STRUCTURE OF PIXEL BUFFER
INCLUDE MAXBYT           8 PROC DEFINING MAXIMUM BYTE
INCLUDE MAXICE           8 PROC DEFINING MAXIMUM INTEGER CHAR EQUIVALENT
INCLUDE MAXINT           8 PROC DEFINING MAXIMUM INTEGER

C
C
C LOCAL DECLARATIONS
C -----
C
      INTEGER MPXSHA(1)      8 ARGUMENT
      INTEGER MPXRAM(1)      8 ARGUMENT
      INTEGER ICOEFO         8 ARGUMENT
      INTEGER ICOEFI         8 ARGUMENT
      INTEGER NI4ONS         8 NUMBER OF INTEGERS REQUIRED FOR PIXEL BINS
      INTEGER ICASE          8 TYPE OF BIN 'BYT'=1, 'CHR'=2, 'INT'=3
      INTEGER LENGTH        8 LENGTH OF PIXEL BUFFER IN INTEGERS
      INTEGER K              8 NUMBER OF SAMPLE TO SHARPEN
      INTEGER IPIX          8 VALUE OF PIXEL TWO BEFORE KPIX
      INTEGER JPIX          8 VALUE OF PIXEL ONE BEFORE KPIX
      INTEGER KPIX          8 VALUE OF PIXEL TO BE SHARPENED
      INTEGER LPIX          8 VALUE OF PIXEL ONE AFTER KPIX
      INTEGER MPIX          8 VALUE OF PIXEL TWO AFTER KPIX
      INTEGER ISHARP        8 SHARPENED VALUE

C
C
C PROCEDURE
C -----
C
C COPY MPXRAM TO MPXSHA
C
      IF (MPXRAM(PXBINT).NE.'BYT') 00 TO 220
      NI4ONS=NI4NB(MPXRAM(PXHBIN))
      ICASE=1
      00 TO 260
220 IF (MPXRAM(PXBINT).NE.'CHR') 00 TO 230
      NI4ONS=NI4NC(MPXRAM(PXHBIN))
      ICASE=2
      00 TO 260
230 IF (MPXRAM(PXBINT).NE.'INT') 00 TO 240
      NI4ONS=MPXRAM(PXHBIN)
      ICASE=3
      00 TO 260
240 CALL MODFATL ('SHASAM DETECTS UNKNOWN BIN TYPE'. , ,
      1 COS4CS(MPXRAM(PXBINT),1,4), ' IN BUFFER')
      00 TO 900
260 LENGTH=PXBSINS-1+NI4ONS
      CALL MOVIST (MPXSHA,1,LENGTH,MPXRAM,1,LENGTH,0)

C
C
C LOCATE EXTREMA PIXELS IN MPXRAM BUFFER. SHARPEN THEM AND
C STORE SHARPENED VALUES IN MPXSHA BUFFER
C
      00 TO (270,280,290),ICASE
270 CALL SHARPNIOETBYT,PUTBYT,MAXBYT)
      00 TO 900

```



DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

SHASAM  
003

200 CALL SHARPN(GETICE,PUTICE,MAXICE)  
GO TO 900  
200 CALL SHARPN(GETINT,PUTINT,MAXINT)

C  
C  
C 900 RETURN

C  
C  
C SUBROUTINE SHARPN (                    0 SHARPEN EXTREMA PIXELS  
1 GETBIN.        0 NAME OF SUBROUTINE TO GET A BIN GETBYT/GETICE/GETINT  
1 PUTBIN.        0 NAME OF SUBROUTINE TO PUT A BIN PUTBYT/PUTICE/PUTINT  
1 MAXBIN)        0 MAXIMUM VALUE FOR A BIN MAXBYT/MAXICE/MAXINT

C  
C  
C INITIALIZE SAMPLE NUMBER AND SLOPE STATE

K=MPXRAW(PXBINS)+1    0 FILTER CAN'T EXTEND OUTSIDE DATAK  
CALL GETBIN(KPIX,MPXRAW(PXBINS),K)  
CALL GETBIN(LPIX,MPXRAW(PXBINS),K+1)  
IF (LPIX.LT.KPIX) GO TO 760

C  
C SLOPE STATE IS POSITIVE

C  
C 750 K=K+1  
IF (K.GT.MPXRAW(PXBINS)-2) GO TO 770  
KPIX=LPIX  
CALL GETBIN (LPIX,MPXRAW(PXBINS),K+1)  
IF (LPIX.GE.KPIX) GO TO 750

C  
C  
C SLOPE STATE CHANGES FROM POSITIVE TO NEGATIVE--SHARPEN UP

C  
C  
C CALL GETBIN (IPIX,MPXRAW(PXBINS),K-2)  
CALL GETBIN (JPIX,MPXRAW(PXBINS),K-1)  
CALL GETBIN (MPIX,MPXRAW(PXBINS),K+2)  
ISHARP=KPIX+(IABS(ICOEFO\*IPIX - ICOEFO\*MPIX + ICOEF1\*JPIX -  
1 ICOEF1\*LPIX) + 2\*\*8)/2\*\*12  
CALL PUTBIN (MPXSHA(PXBINS),K,MIN0(ISHARP,MAXBIN))

C  
C  
C SLOPE STATE IS NEGATIVE

C  
C 760 K=K+1  
IF (K.GT.MPXRAW(PXBINS)-2) GO TO 770  
KPIX=LPIX  
CALL GETBIN (LPIX,MPXRAW(PXBINS),K+1)  
IF (LPIX.LE.KPIX) GO TO 760

C  
C  
C SLOPE STATE CHANGES FROM NEGATIVE TO POSITIVE--SHARPEN DOWN

C  
C  
C CALL GETBIN (IPIX,MPXRAW(PXBINS),K-2)  
CALL GETBIN (JPIX,MPXRAW(PXBINS),K-1)  
CALL GETBIN (MPIX,MPXRAW(PXBINS),K+2)  
ISHARP=KPIX-(IABS(ICOEFO\*IPIX - ICOEFO\*MPIX + ICOEF1\*JPIX -  
1 ICOEF1\*LPIX) + 2\*\*8)/2\*\*12

**DAN PACKAGE APPENDIX H  
UTILITY ROUTINES**

**SHASAN  
004**

CALL PUTBIN (HPXSHA(PXBINS).K.MAXO(ISHARP.0))  
GO TO 750  
770 RETURN  
C  
C  
END

C - 5

**SHFTDC**  
**001**

• (E H SCHLOSSER)

```

S(00) . D-BANK
LEFT      LSSC      A3.0      . LEFT SHIFT INSTRUCTION
RIGHT     SSC       A3.0      . RIGHT SHIFT INSTRUCTION
          END

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

SPANS  
001

```

SUBROUTINE SPANS( 8 ENABLE/DISABLE SPANNING FOR UNIT 5
  I NSPAN) 8 MAXIMUM NUMBER OF LINES TO SPAN FOR CURRENT COMMAND
-----
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      03/08/75      ORIGINAL CODE IN READS
C      E H SCHLOSSER      LEC      02/14/79      REVISE & MAKE SEPARATE SUBROUTINE
C
C
C METHOD
C -----
C
C      STORE NSPAN IN LABELLED COMMON AND ADJUST PADDING OF CURRENT IMAGE.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      INTEGER LENCST      8 LENGTH OF CHARACTER STRING
C      PUTCHR      8 PUT CHARACTER INTO CHARACTER STRING
C
C
C EXCEPTIONS
C -----
C
C      1. VALUES OF NSPAN OUTSIDE THE RANGE FROM 0 TO 32 GENERATE A FATAL ERROR.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMBUS.LIST      8 POINTERS/FLAGS/BUFFER FOR UNIT 5
C
C
C LOCAL DECLARATIONS
C -----
C
C      NONE.
C
C
C PROCEDURE
C -----
C
C
      IF((NSPAN.GE.0).AND.(NSPAN.LE.32)) GO TO 120
      CALL MDFATL( 'NSPAN OUT OF RANGE IN SPANS')

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

SPANS  
002

```
      GO TO 900
120 LUSSPN=NSPAN
    LUSHAX=LENCST(LUSING,LUSHAX)      & ELIMINATE TRAILING BLANK. IF ANY
    CALL PUTCHR(LUSING,(LUSHAX+1),    ' ')
    IF(LUSSPN.EQ.0) LUSHAX=LUSHAX+1    & PAD WITH BLANK ONLY IF NOT SPANNING
C
C
C DONE
C
900 RETURN
    END
```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**SPLIT  
001**

SUBROUTINE SPLIT: 8 SPLIT REAL INTO SIGN, INTEGER, DECIMAL  
I ARG. 8 REAL NUMBER TO BE SPLIT  
O ISGN. 8 SIGN OF ARG ('+' OR '-')  
O INTEOR. 8 ABSOLUTE VALUE OF INTEGER PART  
O DECINL) 8 ABSOLUTE VALUE OF DECIMAL PART  
-----

C  
C  
C (END)  
C

ISGN='+'  
IF(ARG.LT.0.) ISGN='-'  
DECINL=ABS(ARG)  
INTEOR=IFIX(DECINL)  
DECINL=ABS(DECINL-FLOAT(INTEOR))  
IF(INTEOR.GT.999) DECINL=DECINL+1. 8 FLAG OVERFLOW!  
RETURN  
END

REPRODUCED FROM  
OF HIGH QUALITY

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

SREADS  
001

SUBROUTINE SREADS    & SPANNED READ OF UNIT 5 (ONLY CALLED BY OETS.. SERIES)

```

C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      03/08/75      ORIGINAL CODE IN READS
C      E H SCHLOSSER      LEC      02/14/79      REVISE & MAKE SEPARATE SUBROUTINE
C
C
C METHOD
C -----
C
C      IF NEXT FIELD MUST BE SPANNED. THEN SPAN IT BY DOING THE FOLLOWING:
C          DECREMENT & SAVE LUSSPN
C          CALL READS TO FILL IMAGE BUFFER (IT ZEROS LUSSPN!!)
C          RESTORE LUSSPN
C          REMOVE TRAILING BLANK(S) FROM IMAGE BUFFER
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      READS          & FILL IMAGE BUFFER FROM UNIT 5 (ZEROS LUSSPN & PADS WITH BLANK
C      INTEGER LENCST  & LENGTH OF CHARACTER STRING
C
C
C EXCEPTIONS
C -----
C
C      1. ANY NON-BLANK I/O STATUS ON UNIT 5 TERMINATES SPANNING.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMLUS.LIST      & POINTERS/FLAGS/BUFFER FOR UNIT 5
C      INCLUDE NULCST.LIST      & DEFINE NULL CHARACTER STRING
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER NEWSPN          & TEMP LOCATION TO SAVE LUSSPN
C      INTEGER LSSTAT          & I/O STATUS FROM READS (SAME AS LUSIOS)
C      INTEGER MSGBAD(7)      //  STATUS WHEN SPANNING COMMAND'.NULCST/
C
C
C PROCEDURE

```

OAM PACKAGE APPENDIX N  
UTILITY ROUTINES

8READS  
002

C -----  
C  
C

```

      IF(LUSSPN.LE.0) GO TO 900      & SPANNING NOT ALLOWED
      IF(LUSLOC+LUSLEN.NE.LUSMAX) GO TO 900      & DON'T SPAN NOW
      NEWSPN=LUSSPN-1
      IF(LUSIOS.EQ.' ') CALL READS(LSSTAT,  NULCST)
      IF(LUSIOS.EQ.' ') GO TO 120
      CALL MOVCS1(HS0BAD.(1).(3),  LUSIOS.(1).(3),' ')
      CALL NOMARN(  HS0BAD)
      NEWSPN=0
120  LUSSPN=NEWSPN
      LUSMAX=LENCST(LUSIMO.LUSMAX)      & ELIMINATE TRAILING BLANK FROM IMAGE
C
C
C DONE
C
900 RETURN
      END

```



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

SSP%  
001

```

      SUBROUTINE SSPRI 8 COMPUTE SUMS & SUMS OF PRODUCTS
      I TATA.          8 SINGLE PRECISION DATA MATRIX
      I NOBS.          8 NUMBER OF OBSERVATIONS (DATA ROWS)
      I NVAR.          8 NUMBER OF VARIABLES (DATA COLUMNS)
      0 SUM.           8 SINGLE PRECISION VECTOR OF SUMS
      0 SPROD.         8 SINGLE PRECISION MATRIX OF SUMS OF PRODUCTS
      I NRCDIM)        8 NUMBER OF ROWS & COLUMNS DIMENSIONED IN SPROD
      -----
C
C
C (E H SCHLOSSER)
C
C
      DIMENSION TATA(NOBS,NVAR)
      DIMENSION SUM(1),SPROD(NRCDIM,NRCDIM)
      CALL TRACE
      DATA NITER /1/      8-8-8 TEMP PATCH
C
C
C
C INITIALIZE SUMS & SUMS OF PRODUCTS
C
      DO 110 NR=1,NVAR
      SUM(NR)=0.
      DO 110 NC=1,NVAR
      110 SPROD(NR,NC)=0.
C
C
C COMPUTE SUMS & UPPER TRIANGULAR SUMS OF PRODUCTS
C
      DO 140 I=1,NITER      8-8-8 TEMP PATCH
      DO 140 NOB=1,NOBS
      DO 130 NR=1,NVAR
      SUM(NR)=SUM(NR)+TATA(NOB,NR)
      DO 120 NC=NR,NVAR
      120 SPROD(NR,NC)=SPROD(NR,NC)+TATA(NOB,NR)*TATA(NOB,NC)
      130 CONTINUE
      140 CONTINUE
      NOBS=NOBS+NITER      8-8-8 TEMP PATCH
C
C
C COPY UPPER TRIANGULAR SUMS OF PRODUCTS TO LOWER TRIANGULAR
C
      DO 230 NR=1,NVAR
      DO 220 NC=NR,NVAR
      220 SPROD(NC,NR)=SPROD(NR,NC)
      230 CONTINUE
C
      RETURN
      ENTRY SSPRI(ITER)      8-8-8 TEMP PATCH
      NITER=ITER      8-8-8 TEMP PATCH
      RETURN      8-8-8 TEMP PATCH
      END

```

```

SUBROUTINE STREGG  & STORE REGISTRATION PARAMETERS ON UNIT 0
-----
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      06/01/73      ORIGINAL CODE
C      E H SCHLOSSER      LEC      09/14/78      REORGANIZE NAMELIST SPECS
C      E H SCHLOSSER      LEC      12/12/79      CLOSE & REWIND 0 FOR MULTIPLE ADJUST
C
C METHOD
C -----
C
C      WRITE REGISTRATION PARAMETERS ON TEMPORARY DISK FILE 0 IN FORTRAN
C      NAMELIST FORMAT. (NAMELIST IS USED SO THAT THE FILE MAY BE EXAMINED
C      WITH THE SYSTEM TEXT EDITOR FOR DEBUGGING PURPOSES )
C
C MACHINE-DEPENDENT CODE
C -----
C
C      DIMENSION & FORMAT SPECIFICATIONS ASSUME 8 CHARACTERS PER WORD.
C      NAMELIST I/O.
C
C EXTERNAL REFERENCES
C -----
C
C      MDWARN      & PRINT/COUNT/LOG 'WARNING' DIAGNOSTIC MESSAGE
C      MDFATL      & PRINT/COUNT/LOG 'FATAL ERROR' DIAGNOSTIC MESSAGE
C      CLOSE      & UNIVAC FORTRAN V SYSTEM ROUTINE; CLOSE & REWIND FILE
C
C EXCEPTIONS
C -----
C
C      1. IF THE CONTROL NETWORK EFFECTIVELY COVERS LESS THAN 50% OF THE SCENE,
C          THEN A 'WARNING' IS ISSUED.
C
C      2. IF THE ROOT-MEAN-SQUARE (RMS) ERROR OF THE ADJUSTMENT IS GREATER THAN
C          400 METERS, THEN A 'FATAL ERROR' IS ISSUED AND THE REGISTRATION
C          PARAMETERS ARE NOT STORED.
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KONNER.LIST      & COMMON ERTS SCENE PARAMETERS
C      INCLUDE KONFIT.LIST      & COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C      INCLUDE LSTLUB.LIST      & NAMELIST SPECIFICATIONS FOR UNIT 0
C
C LOCAL DECLARATIONS
C -----

```

**STRECH**  
**002**

**N-374**

SUBROUTINE SUBWIN:    : BREAK WINDOW INTO SUBWINDOWS BASED ON PRIMARY TICKS  
:  
: RLTOHW.   : REAL OUTPUT WINDOW PACKET (COORD SYSTEM OF PRIMARY TICKS)  
: MXSUBW.   : MAXIMUM NUMBER OF SUBWINDOWS  
: NANSUB)   : NAME OF SUBROUTINE TO GENERATE SUBWINDOW MAPS  
:-----

C  
C  
C  
C HISTORY  
C -----

C       E M SCHLOSSER       LEC       07/22/73       ORIGINAL CODE (INTERNAL SUB)  
C       E M SCHLOSSER       LEC       01/10/79       CONVERT TO EXTERNAL SUBROUTINE

C  
C  
C METHOD  
C -----

C       TRANSFORM ENVELOPE OF OUTPUT WINDOW FROM ITS OWN COORDINATE SYSTEM INTO  
C       PRIMARY TICK UNITS ( AN INTEGER COORDINATE SYSTEM WITH AN INTERVAL OF 1  
C       BETWEEN PRIMARY TICKS). IF THIS ENVELOPE CONTAINS MORE TICKS (SUB-WINDOW  
C       ORIGINS) THAN REQUESTED BY MXSUBW. THEN SHRINK THE ENVELOPE TO ELIMINATE  
C       THE ORIGINS OF FRACTIONAL SUB-WINDOWS AT THE EDGES OF THE ENVELOPE.  
C       COMPUTE ALL TICKS (SUB-WINDOW ORIGINS) WITHIN THE REMAINING ENVELOPE.  
C       GENERATING A MAP AT EACH SUB-WINDOW ORIGIN.

C       SUB-WINDOW ORIGINS AND VERTICES MUST BE STORED IN THE OUTPUT WINDOW PACKET  
C       TO GENERATE MAPS FOR THESE SUB-WINDOWS. THEREFORE, THE ORIGINAL CONTENTS  
C       OF THIS PORTION OF THE OUTPUT WINDOW PACKET ARE SAVED ON ENTRY AND RESTORE  
C       ON EXIT FROM THIS ROUTINE.

C  
C  
C MACHINE-DEPENDENT CODE  
C -----

C       BOOL.

C  
C  
C EXTERNAL REFERENCES  
C -----

C       CALWIN       : CAL'BRATE OUTPUT SUB-WINDOW ENVELOPES & VERTICES

C  
C  
C EXCEPTIONS  
C -----

C       1. NO CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

C  
C  
C GLOBAL DECLARATIONS  
C -----

C       INCLUDE WINDEF.LIST       : DEFINE STRUCTURE OF WINDOW PACKETS  
C       INCLUDE KOMINW.LIST       : COMMON INPUT WINDOW PACKETS  
C       INCLUDE KONOWW.LIST       : COMMON OUTPUT WINDOW PACKETS

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

SUBMIN  
002

```

C
C
C LOCAL DECLARATIONS
C -----
C
  REAL RLTONH(2,1)      2 ARGUMENT
  PARAMETER NVER2=NVER+2  2 NUMBER OF NODES FROM RLTONH TO SAVE IN SAVONH
  REAL SAVONH(2,NVER2)  2 ARRAY FOR TEMP SAVE OF PART OF RLTONH
  INTEGER NSUB          2 NUMBER OF SUB-WINDOWS REMAINING TO GENERATE
  INTEGER KSYORI        2 COORDINATE SYSTEM OF OUTPUT WINDOW ORIGIN
  INTEGER KSYVER        2 COORDINATE SYSTEM OF OUTPUT WINDOW VERTICES
  INTEGER MAX,NOD       2 WINDOW PACKET AXIS,NODE
  INTEGER MTIC1,MTIC2    2 COORDINATES OF TICK IN PRIMARY TICK UNITS
  INTEGER MINT1,MINT2    2 OUTPUT ENVELOPE MINIMUMS IN PRIMARY TICK UNITS
  INTEGER MAXT1,MAXT2    2 OUTPUT ENVELOPE MAXIMUMS IN PRIMARY TICK UNITS
  REAL TIMIN,TSMIN      2 OUTPUT ENVELOPE MINIMUMS IN PRIMARY TICK UNITS
  REAL TIMAX,TSMAX      2 OUTPUT ENVELOPE MAXIMUMS IN PRIMARY TICK UNITS

C
C
C PROCEDURE
C -----
C
C CALL TRACE
C
C
C SAVE OUTPUT WINDOW
C
  KSYORI=KSYONH(NORI0)
  KSYVER=KSYONH(NVER)
  DO 100 NOD=NHEAD,NVER2
    DO 130 MAX=1,2
      SAVONH(MAX,NOD)=RLTONH(MAX,1,NOD)
  130 CONTINUE
  100 CONTINUE

C
C
C INITIALIZE OUTPUT SUB-WINDOW
C
  KSYONH(NORI0)=KSYONH(MTIC)      2 PRIMARY TICK COORDINATE SYSTEM
  KSYONH(NVER)=KSYONH(MTIC)
  RLTONH(NUSED,NHEAD)=BCOL(NVER2)  2 INTEGER IN REAL ARRAY!!!!
  RLTONH(1,NVER+0)=0.             2 CLOSING VERTEX
  RLTONH(2,NVER+0)=0.
  RLTONH(1,NVER+1)=RLTONH(1,MTIC)  2 1ST VERTEX IS PRIMARY TICK INTERVAL
  RLTONH(2,NVER+1)=RLTONH(2,MTIC)
  RLTONH(1,NVER+2)=0.             2 2ND VERTEX IS 0
  RLTONH(2,NVER+2)=0.

C
C
C COMPUTE OUTPUT ENVELOPE IN TERMS OF PRIMARY TICK UNITS
C
  TIMIN=SAVONH(1,NMIN)/SAVONH(1,MTIC)*0.5000001
  TIMAX=SAVONH(1,NMAX)/SAVONH(1,MTIC)*0.4999999
  TSMIN=SAVONH(2,NMIN)/SAVONH(2,MTIC)*0.5000001
  TSMAX=SAVONH(2,NMAX)/SAVONH(2,MTIC)*0.4999999

```

**SUBMIT  
003**

**M-377**

SUBROUTINE TRACE( 8 CONDITIONAL TRACE OF SUBROUTINE CALL  
I NAMENT) 8 NAME OF ENTRY POINT (OPTIONAL 8 CHAR ARGUMENT)  
-----

HISTORY  
-----

E H SCHLOSSER	LEC	06/18/74	ORIGINAL CODE
E H SCHLOSSER	LEC	11/09/79	SPACE BETWEEN SUB NAME & ENTRY NAME
E H SCHLOSSER	LEC	01/07/80	KOMXQT MACRO & END TRACE QUERY

METHOD  
-----

```

DETERMINE NUMBER OF ARGUMENTS (0 OR 1)
IF MTRACE IN KOMXQT LABELLED COMMON IS NOT ZERO:
    PRINT THE CALLING SUBROUTINE NAME (AND ENTRY POINT NAME)
IF MTRACE IN KOMXQT IS ZERO (DEMAND MODE):
    IF MADLEY IN KOMXQT IS ZERO:
        MTRACE := MOD(MTRACE+1,64)
    IF MTRACE IS ZERO:
        ASK: 'END TRACE?'
        IF 1ST CHAR OF RESPONSE (<> 'Y'):
            MTRACE := 1

```

MACHINE-DEPENDENT CODE  
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 8-BIT  
FIELDATA CHARACTERS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
THIS ROUTINE EXTRACTS THE NAME OF THE CALLING ROUTINE FROM THE UNIVAC  
FORTRAN V WALKBACK PACKET.  
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.  
DIFFERENT COMPILERS (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

EXTERNAL REFERENCES  
-----

ER PRINTS	8 EXEC REQUEST: PRINT IMAGE
ER TREADS	8 EXEC REQUEST: PRINT & READ

EXCEPTIONS  
-----

1. RESULTS ARE UNDEFINED IF TRACE IS CALLED WITH MORE THAN 1 ARGUMENT.
2. RESULTS ARE UNDEFINED IF USER RESPONSE TO AN 'END TRACE?' QUERY IS MORE THAN 84 CHARACTERS LONG.

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

TRACE/DAN  
002

. GLOBAL DECLARATIONS  
-----

AXRS  
KONXQT

. STANDARD UNIVAC 1100 REGISTER MNEMONICS  
. COMMON PROGRAM EXECUTION SWITCHES

. LOCAL DECLARATIONS  
-----

S(00) . D-BANK

TRABUF	RES	14	. 14 WORD BUFFER HOLDS 84 6-BIT CHARS
SPACES			
ASK			. END TRACE?>>>
PF	FORM	12.6.10	. PKT FORM (BITS): ADVANCE.WDS.BUFOUT ADDR
TRAPKT	PF	1.3.0	. TRACE PKT: ADVANCE 1. LENGTH 1 WD. DUMMY
ASKPKT	PF	1.2.0	. ASK PKT: ADVANCE 1. LENGTH 2 WDS. DUMMY
	RES	1	. ASK PKT: EOF ADDR. BUFIN ADDR

. PROCEDURE  
-----

S(01) . I-BANK

TRACE	LA	A2.SPACES	. BLANK OUT ARGUMENT FROM PREVIOUS CALL
	TZ.S1	1.X11	. WALK-BACK WORD?
	J	NOARG	. NO. ITS AN INSTRUCTION!
ONEARG	LA	A2.*0.X11	. ONE ARG -- GET IT
	AX.XU	X11.1	. POINT X11 TO WALK-BACK WD
NOARG	TNZ	MTRACE	. CHECK TRACE SWITCH
	J	1.X11	. SWITCH IS OFF -- RETURN
TRACING	LA	A3.0.X11	. WALKBACK WD (M2=ADDR OF SUBR NAME)
	LA	A1.0.A	. GET SUBROUTINE NAME
	LA	A0.TRAPKT	. X1 := PRINT SPECS
	SA	A1.TRABUF+0	. PUT SUB NAME IN 1ST WD OF BUFFER
	LA	A3.SPACES	
	DSC	A2.0	. SHIFT ARG 1 SPACE TO RIGHT
	DS	A2.TRABUF+1	. PUT SHIFTED ARG IN NEXT 2 WDS OF BUFFER
	LXM	A0.(TRABUF)	. XM := ADDR OF BUFFER
	ER	PRINTS	. PRINT BUFFER
	TZ	MADLEV	. CHECK NESTED ADD LEVEL
	J	1.X11	. LEVEL IS NON-ZERO -- RETURN
MADLEV0	TZ	MATCH	. CHECK MATCH SWITCH
	J	1.X11	. SWITCH IS ON -- RETURN
DEMAND	LA	A2.MTRACE	. A2 := MTRACE
	AA.U	A2.1	. A2 := A2+1
	SA	A2.MTRACE	. MTRACE := A2
	SBL	A2.0	. A2 := A2/84
	JZ	A2.1.X11	. RETURN IF NEW MTRACE < 84



DAM PACKAGE APPENDIX M  
UTILITY ROUTINES

TRACE/DAM  
003

ASKUSER	SA	A2.MTRACE	. CHANGE MTRACE TO 1
	LA	A0.(ASK)	.
	LA	A1.(TRABUF)	.
	SA.M2	A0.ASKPKT+0	.
	LXI	A1.(EOF)	.
	SA	A1.ASKPKT+1	.
	LA	A0.(ASKPKT)	. ASK USER IF HE WANTS ...
	ER	TREADS	. ... TO END TRACE
	LA.S1	A1.TRABUF	. A1 := FIRST CHAR OF ANSWER
	TE.U	A1.'88888Y'	. SKIP NEXT INSTR IF = 'Y'
	J	1.X11	. NOT = 'Y' -- KEEP TRACING
TURNOFF	SZ	MTRACE	. TURN OFF TRACE
EOF	J	1.X11	. RETURN
	END		

**TRECVR**  
**001**

**N-301**

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

TRECVR  
002

```

      INTEGER NOVDIR          & DIRECTION OF READ

C
C
C PROCEDURE
C -----
C
C      CALL TRACE
C
C READ FORWARD ONE RECORD
C
      NOVDIR=1
      CALL ERTWAT(2000)
      CALL R3TREC(MPXBUF,(NHIBF),ITSTAT,NOVDIR,200,300,80)
C
C
C CHECK STATUS OF READ
C
      IF (ITSTAT.EQ.'EOF') GO TO 100
      IF (ITSTAT.EQ.' ') GO TO 200
C
C
C ANY OTHER STATUS IS BAD FILE
C
      ITSTAT='BADF'
      GO TO 900
C
C
C IF 'EOF', READ BACK OVER RECORD
C
      100 CALL ERTWAT(2000) & REVERSE TAPE GENTLY
      NOVDIR=-1
      CALL R3TREC(MPXBUF,(NHIBF),ITSTAT, NOVDIR,200,300,80)
C
C
C SET RECORD IN BUFFER TO LAST RECORD IN FILE
C
      LU3RBF=LU3RHI(LU3VOL)
C
C
C IF LAST RECORD IS ONE REQUESTED. RETURN 'BAD RECORD' STATUS
C
      IF (LU3RBF.EQ.NSAREC) GO TO 150
C
C
C ELSE, TRY AGAIN TO LOCATE RECORD
C
      ITSTAT=' '
      GO TO 900
      150 ITSTAT='BADR'
      GO TO 900
C
C
C IF NORMAL COMPLETION. CHECK RECORD NUMBER OF RECORD READ
C
      200 CALL OCTQBY(LU3RBF, MPXBUF(PXBINS),1)

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

TRECVR  
003

```
C
C
C IF RECORD AFTER REQUESTED RECORD IS FOUND. RETURN 'BAD RECORD'
C STATUS FOR REQUESTED RECORD
C
    IF (MSAREC.EQ.LU3RBF-1) GO TO 250
C
C ELSE. TRY AGAIN TO LOCATE RECORD
C
    ITSTAT=' '
    GO TO 900
250 ITSTAT='BADR'
900 RETURN
END
```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

TSHAPS  
001

```

SUBROUTINE TSHAP3( 8 INITIALIZE TAPE SHAP FOR UNIT 3
0 ITSTAT .      8 TAPE SHAP STATUS:
C              . . - OK
C              'ERR' - ERROR
C
      1 NTPVOL)      8 REQUESTED TAPE VOLUME NUMBER
-----
C
C
C HISTORY
C -----
C
C      MARY TOMPKINS      LEC      08/28/79      REQUIREMENTS
C      MARY TOMPKINS      LEC      10/02/79      ALGORITHM DESIGN
C      MARY TOMPKINS      LEC      10/02/79      ALGORITHM CODE
C
C METHOD
C -----
C
C      IF NTPVOL < 1 OR > LU3VHI ISSUE NOTE AND RETURN. IF LU3REL(NTPVOL)
C      = ' ' .LOCATE LAST DEFINED TAPE VOLUME AND SHAP.FORWARD UNTIL
C      LU3REL(NDFFVL) <> ' ' ELSE ISSUE NOTE AND RETURN.LUPKT(NTPVOL) =
C      LU3REL(3) . CALL ERTSWP TO SHAP TAPE. READ TAPE DIRECTORY IF I/O
C      ERROR ISSUE NOTE RETURN ELSE EXTRACT/DECODE INFORMATION TO
C      VERIFY THAT TAPE IS CORRECT VOLUME FROM SET. IF NTPVOL + 1 <= LU3VHI
C      AND LU3REL(NTPVOL + 1) = ' ' CALL FLINFO. LU3REL(NTPVOL + 1) = LUPKT(13)
C      LU3RBF = LU3RLO(LU3VOL) - 1.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      UTILIZES EXEC 8 ERTSHAPS
C
C EXTERNAL REFERENCES
C -----
C
C      MONOTE      8 PRINT/LOG 'NOTE' MESSAGES
C      KHR4IN      8 ENCODE CHARACTER STRING FROM INTEGER
C      FLINFO      8 RETRIEVE FACILITIES ASSIGNMENT INFORMATION
C      R3TREC      8 READ ONE DATA RECORD FROM TAPE
C      ERTSWP      8 CLOSE CURRENT REEL OF TAPE FILE & REQUEST LOADING OF OTHER
C      GETBYT      8 GET NON-NEGATIVE INTEGER FROM BYTE IN BYTE STRING
C      CST4AS      8 CHARACTER STRING FOR ASCII BYTE STRING
C      GETCHR      8 GET CHARACTER FROM CHARACTER STRING
C      MOVCHR      8 MOVE CHARACTER STRING
C      DCODE      8 DECODE NUMERIC CHARACTER STRING
C      DOUBLE PRECISION CBS4CS 8 VARIABLE LENGTH CST FOR FIXED LENGTH CST
C      INTEGER ND4NI      8 4 BYTES FOR 4 INTEGERS
C
C
C EXCEPTIONS
C -----
C

```

**TSHAP3**  
**002**

C	NTPVOL<1 OR >LU3VHI	QUEUED MONOTE
C	NO REEL NO. FOR NTPVOL	QUEUED MONOTE
C	FIRST RECORD TYPE NOT DIRECTORY	QUEUED MONOTE
C	WRONG VOL. NO. IN DIRECTORY	QUEUED MONOTE
C	WRONG SCENE IN DIRECTORY	QUEUED MONOTE
C	WRONG LU3SEQ IN DIRECTORY	QUEUED MONOTE
C	WRONG LU3REF IN DIRECTORY	QUEUED MONOTE
C	'BADF' WHEN READING DIRECTORY	QUEUED MONOTE
C	'BADR' WHEN READING DIRECTORY	QUEUED MONOTE
C	'LOST' WHEN READING DIRECTORY	QUEUED MONOTE
C	'EOF' WHEN READING DIRECTORY	QUEUED MONOTE

## GLOBAL DECLARATIONS

**f** 

```

C
C
C  LOCAL DECLARATIONS
C  -----
C

```

## C. PROCEDURE

**E** .....  
.....

### CALL TRACE

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**TSHAPS  
003**

```

      IFUNC = 2      2 UNIVAC CODE FOR SHAPPING TO PRESCRIBED REEL
      NTYPE = ' '
      NUPKT(1) = LU3PKT(1)
      NUPKT(2) = LU3PKT(2)
C
C
C INITILIZE STATUS TO WORST CASE
C
      ITSTAT = 'ERR'
C
C
C VOLUME WITHIN VALID RANGE ?
C
      IF( (NTPVOL.LE.0).OR.(NTPVOL.GT.LU3VM1) ) CALL MONOTE(
        * '( TAPE VOLUME OUT OF RANGE)' * )
      IF( (NTPVOL.LE.0.).OR.(NTPVOL.GT.LU3VM1) ) GO TO 900
C
C
C DETERMINE IF TAPE NUMBER IS DEFINED
C
      IF(LU3REL(NTPVOL).NE.' ') GO TO 140
C
C
C LOCATE THE LAST PREVIOUSLY DEFINED TAPE
C
      MINVOL = NTPVOL - 1 2 VOLUME BEFORE REQUESTED
      DO 100 NUM = MINVOL,1,-1
        NDEFVL = NUM      2 NEAREST PREVIOUS DEFINED TAPE VOL.
        IF(LU3REL(NUM).NE.' ') GO TO 120
      100 CONTINUE
C
      CALL MONOTE( '(ERROR IN TAPE ASSIONMENT)' * )
      GO TO 900
C
C
C SHAP TAPES TO ESTABLISH NEXT REEL NUMBER
C
      120 DO 130 NUMVOL = NDEFVL,MINVOL
        NUPKT(3) = LU3REL(NUMVOL) 2 REEL NO. TO SHAP TO
        CALL ERTSHP(NUPKT,IFUNC) 2 SHAP TAPES
        LU3VOL = NUMVOL      2 CURRENT TAPE VOLUME
        CALL FLINFO(IDFILE, LU3PKT,'00') 2 UPDATE PACKET
        LU3FID(FIDCRL) = IDFILE(FIDCRL)
        LU3FID(FIDNRL) = IDFILE(FIDNRL)
        IF(IDFILE(7).EQ.' ') CALL MONOTE(
          * '(TAPE NOT ASSIGNED)' * )
        IF(IDFILE(7).EQ.' ') GO TO 900
      130 LU3REL(LU3VOL + 1) = IDFILE(7) 2 NEXT REEL NO
C
C
C REQUESTED TAPE REEL NUMBER AVAILABLE
C
      140 NUPKT(3) = LU3REL(NTPVOL)
      CALL ERTSHP(NUPKT,IFUNC) 2 SHAP TAPES
      LU3VOL = NTPVOL
      IF(NTPVOL.LT.LU3VM1) CALL FLINFO(IDFILE, LU3PKT,'00')

```

DAM PACKAGE APPENDIX H  
UTILITY ROUTINES

TSHAPS  
004

```

      IF( (NTPVOL.LT.LU3VHI).AND.(IDFILE(7).NE.' ') )
      = LU3REL(LU3VOL + 1) = IDFILE(7)  & NEXT REEL NO.
C
C
C ON A RETURN CALL TO VOL. 1 TAPE DIR. IS NOT CHECKED
C AND RECORDS BEFORE IMAGE ARE READ OVER
C
      IF(NTPVOL.NE.1) GO TO 150
      ITSTAT = ' '
      LU3REF = LU3REL(LU3VOL)
      DO 100 NREC = 1,40
      CALL R3TREC(ITDUF,(RNWITD),ISTAT, 1,200,300,00)
      IF(ISTAT.EQ.'BADF') ITSTAT = 'ERR'
      IF(ISTAT.EQ.'BADF') GO TO 900
      CALL GETBYT(NUMREC, ITDUF,6) & LOCATE IMAGE RECORD
      IF(NUMREC.EQ.0399) GO TO 900
100 CONTINUE
      ITSTAT = 'ERR'
      CALL MDATL( 'IMAGE REC. NOT LOCATED ON RETURN CALL TO VOL. 1')
      GO TO 900
C
C
C READ TAPE DIRECTORY TO DETERMINE TAPE CORRESPONDENCE
C
150 CALL R3TREC(ITDUF,(RNWITD),ISTAT, 1,200,300,00)
      IF(ISTAT.NE.' ') CALL MONOTE(
      * '(I/O STATUS IS', ' ',CBS4CS(ISTAT,1,4), ' ON CONTINUATION TAPE)' )
      IF(ISTAT.NE.' ' .AND. ISTAT.NE.'BADR') GO TO 900
      CALL GETBYT(NUMREC, ITDUF,6) & VERIFY RECORD IS DIRECTORY
      IF(NUMREC.NE.9) CALL MONOTE(
      * '(FIRST RECORD ON REEL ',CBS4CS(LU3REL(LU3VOL),1,6),
      * ' NOT TAPE DIRECTORY)' )
      IF(NUMREC.NE.9) GO TO 900
      CALL GETBYT(NUMSEQ, ITDUF,31) & STORE DATA SEQUENCE
C
C
C TRANSLATE DATA TO CHARACTER STRING
C
      CALL CST4AS(ITDUF, ITDUF.NB4NI(RNWITD))
C
C
C CHECK SENSOR AND TAPE TYPE
C
      CALL GETCHRINSENTRY, ITDUF,9) & 'M' OR 'R'
      IF(SENTRY.NE.'M') CALL MONOTE(
      * '(DATA ON CONTINUATION REEL ',CBS4CS(LU3REL(LU3VOL),1,6),
      * ' NOT MSS)' )
      IF(SENTRY.NE.'M') GO TO 900
      CALL MOVCHRINTYPE,1, ITDUF,11) & 'A' OR 'P'
      CALL MOVCHRINTYPE,2, ITDUF,9) & 'M' OR 'R'
      IF(INTYPE.NE.LU3REF(1)) CALL MONOTE(
      * '(DATA ON CONTINUATION REEL ',CBS4CS(LU3REL(LU3VOL),1,6),
      * ' NOT', ' ',CBS4CS(LU3REF(1),1,4), ')' )
      IF(INTYPE.NE.LU3REF(1)) GO TO 900
C
C

```



DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

TSHAP3  
000

C CHECK DATA SEQUENCE

```
C
  IF(NUMSEQ.EQ.0) NUMSEQ = 'DSQ'
  IF(NUMSEQ.EQ.0377) NUMSEQ = 'SIL'
  IF(NUMSEQ.NE.LU3SEQ(1)) CALL MONOTE(
    = '(DATA ON CONTINUATION REEL *'.CDS4CS(LU3REL(LU3VOL).1.6).
    = ' NOT'. ' .CDS4CS(LU3SEQ(1).1.3).')' )
```

C  
C  
C CHECK VOLUME NUMBER

```
C
  ISTAT = ' '
  CALL DCODE(NUMVOL.REAL.KODTYP, ITDUF.19.1)
  IF(NUMVOL.NE.LU3VOL)KODTYP = 'ERR'
  IF(KODTYP.NE.'IN')CALL MONOTE(
    = '(BAD VOLUME NUMBER ON CONTINUATION REEL *'.
    = 'CDS4CS(LU3REL(LU3VOL).1.6).')' )
  IF(KODTYP.NE.'IN')ISTAT = 'ERR'
```

C  
C  
C COMPARE SCENE ID

```
C
  CALL DCODE(NSCENE(1).REAL.KODTYP, ITDUF.35.1) & MISSION NO.
  IF(NSCENE(1).NE.NERTS(1)) KODTYP = 'ERR'
```

```
C
  IF(KODTYP.EQ.'IN') CALL DCODE(
    = NSCENE(2).REAL.KODTYP, ITDUF.36.4) & DAY SINCE LAUNCH
  IF(NSCENE(2).NE.NERTS(2))KODTYP = 'ERR'
```

```
C
  IF(KODTYP.EQ.'IN')CALL DCODE(
    = NSCENE(3).REAL.KODTYP, ITDUF.40.5) & HOUR.MINUTE.10 SEC
  IF(NSCENE(3).NE.NERTS(3))KODTYP = 'ERR'
```

```
C
  IF(KODTYP.EQ.'ERR')CALL MONOTE(
    = '(BAD SCENE ID ON CONTINUATION REEL'. ' ' .
    = 'CDS4CS(LU3REL(LU3VOL).1.6) )
  IF( (ISTAT.EQ.'ERR').OR.(KODTYP.NE.'IN') ) GO TO 900
```

```
C
  ISTAT = ' '
  LU3RBF = LU3RLO(LU3VOL) - 1
  DO 400 NREC = 1,36
    CALL R3TREC(ITDUF.(RNWITD).ISTAT, 1,200,300,80)
    IF(ISTAT.EQ.'EOF') GO TO 900
```

```
400 CONTINUE
  CALL MONOTE('(EXPECTED EOF NOT FOUND ON CONTINUATION REEL'.
    = 'CDS4CS(LU3REL(LU3VOL).1.6).')' )
  ISTAT = 'ERR'
```

C  
C  
C 900 RETURN

```
C
  END
```

**VALKEY**  
**001**

**N-389**

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

VARSQN  
001

REAL FUNCTION

0                   0 NORMALIZED VARIMAX ROTATION CRITERION  
0 VARSQN(           0 SINGLE PRECISION MATRIX  
1 AMAT.           3 NUMBER OF ROWS & COLUMNS USED  
1 NRUSE,NCUSE.    3 NUMBER OF ROWS & COLUMNS DIMENSIONED  
1 NRDIM,NCDIM.    3 COLUMN NUMBERS OF COLUMNS TO PROCESS  
1 NCIPRO,NCZPRO)   -----

C  
C  
C (E H SCHLOSSER)  
C  
C

DIMENSION AMAT(NRDIM,NCDIM)  
DEFINE AMAT2(NR,NC)=AMAT(NR,NC)\*AMAT(NR,NC)/  
& (AMAT(NR,NCIPRO)\*AMAT(NR,NCIPRO)+AMAT(NR,NCZPRO)\*AMAT(NR,NCZPRO))  
CALL TRACE

C  
C

VARSQN=0.  
NCPRO=NCIPRO  
DO 260 NC=1,2  
SUM4=0.  
SUM2=0.  
DO 240 NR=1,NRUSE  
SUM4=SUM4+AMAT2(NR,NCPRO)\*AMAT2(NR,NCPRO)  
SUM2=SUM2+AMAT2(NR,NCZPRO)

240 CONTINUE  
VARSQN=VARSQN+NRUSE\*SUM4-SUM2\*SUM2  
NCPRO=NCZPRO  
260 CONTINUE  
VARSQN=VARSQN/NRUSE\*NRUSE  
RETURN  
END

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**VARSQU  
001**

```

      REAL FUNCTION
      0
      0 UN-NORMALIZED VARIMAX ROTATION CRITERION
      -VARSQU
      1 AMAT.
      0 SINGLE PRECISION MATRIX
      1 NRUSE,NCUSE.
      0 NUMBER OF ROWS & COLUMNS USED
      1 NRDIM,NCDIM.
      0 NUMBER OF ROWS & COLUMNS DIMENSIONED
      1 NC1PRO,NC2PRO)
      0 COLUMN NUMBERS OF COLUMNS TO PROCESS
      -----
C
C
C (E H SCHLOSSER)
C
C
      DIMENSION AMAT(NRDIM,NCDIM)
      DEFINE AMAT2(NR,NC)=AMAT(NR,NC)*AMAT(NR,NC)
      CALL TRACE
C
C
      VARSQU=0.
      NCPRO=NC1PRO
      DO 200 NC=1,2
      SUM4=0.
      SUM2=0.
      DO 240 NR=1,NRUSE
      SUM4=SUM4+AMAT2(NR,NCPRO)*AMAT2(NR,NCPRO)
      SUM2=SUM2+AMAT2(NR,NCPRO)
240 CONTINUE
      VARSQU=VARSQU+NRUSE*SUM4-SUM2*SUM2
      NCPRO=NC2PRO
200 CONTINUE
      VARSQU=VARSQU/NRUSE*NRUSE
      RETURN
      END

```

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

VERA40  
001

SUBROUTINE VERA40    8 ADJUSTED MSS WINDOW VERTICES FROM GEOGRAPHIC VERTICES  
-----

```

C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      10/07/73      ORIGINAL CODE
C      E H SCHLOSSER      LEC      10/30/79      UPGRADE DOCUMENTATION
C
C
C METHOD
C -----
C
C      TRANSFORM VERTICES IN THE OUTPUT GEOGRAPHIC WINDOW PACKET INTO
C      VERTICES IN THE OUTPUT ADJUSTED MSS WINDOW PACKET.  ALL VERTEX
C      COORDINATES ARE RELATIVE TO THEIR RESPECTIVE WINDOW ORIGINS.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      A40                8 ADJUSTED MSS COORDINATES FOR GEOGRAPHIC COORDINATES
C
C
C EXCEPTIONS
C -----
C
C      NONE.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KONOWH.LIST        8 COMMON OUTPUT WINDOW PACKETS
C      INCLUDE WINDEF.LIST       8 DEFINE STRUCTURE OF WINDOW PACKETS
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER MAXVER        8 MAXIMUM (LAST) VERTEX DEFINED (USED)
C      INTEGER NOO           8 NODE NUMBER
C      REAL ADJLIN,ADJSAM    8 ADJUSTED MSS LINE, SAMPLE
C
C
C PROCEDURE
C -----
C
C      CALL TRACE
C

```

**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**VERA40  
002**

```
C
C INITIALIZE POINTER TO LAST VERTEX USED
C
      MAXVER=0001(0ED0WH(WUSED,WHEAD))      8 INTEGER POINTER IN REAL ARRAY!
      MSA0WH(WUSED,WHEAD)=MAXVER
C
C
C TRANSFORM VERTICES
C
      DO 200 NOD=WVER.MAXVER
        CALL A401
        8      ADJLIN,ADJSAM.
        -      0ED0WH(WLAT,NOD)+0ED0WH(WLAT,WORIG).
        8      0ED0WH(WLON,NOD)+0ED0WH(WLON,WORIG))
        MSA0WH(WLIN,NOD)=IFIX(ADJLIN+.5)-MSA0WH(WLIN,WORIG)
        MSA0WH(WSAM,NOD)=IFIX(ADJSAM+.5)-MSA0WH(WSAM,WORIG)
      200 CONTINUE
C
      RETURN
      END
```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

VERA4P  
001

```

SUBROUTINE VERA4P  & ADJUSTED MSS WINDOW VERTICES FROM PRINT/PLOT VERTICES
-----
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      10/07/73      ORIGINAL CODE
C      E H SCHLOSSER      LEC      10/30/79      UPGRADE DOCUMENTATION
C
C METHOD
C -----
C
C      TRANSFORM VERTICES IN THE OUTPUT PRINT/PLOT WINDOW PACKET INTO
C      VERTICES IN THE OUTPUT ADJUSTED MSS WINDOW PACKET.  ALL VERTEX
C      COORDINATES ARE RELATIVE TO THEIR RESPECTIVE WINDOW ORIGINS.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      A4P      & ADJUSTED MSS COORDINATES FOR PRINT/PLOT DEVICE COORDINATES
C      MDWARN   & PRINT/COUNT/LOG 'WARNING' DIAGNOSTIC MESSAGE
C
C EXCEPTIONS
C -----
C
C      1. IF NEITHER CALSPA NOR CALSCA HAVE BEEN CALLED PREVIOUSLY. THEN THE
C      RESULTS OF VERA4P ARE UNDEFINED.
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMOWH.LIST      & COMMON OUTPUT WINDOW PACKETS
C      INCLUDE WINDEF.LIST      & DEFINE STRUCTURE OF WINDOW PACKETS
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER MAXVER      & MAXIMUM (LAST) VERTEX DEFINED (USED)
C      INTEGER NOD         & NODE NUMBER
C      REAL ADJLIN,ADJSAM  & ADJUSTED MSS LINE. SAMPLE
C
C
C PROCEDURE
C -----
C

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

VERANP  
002

```

CALL TRACE
C
C
C INITIALIZE POINTER TO LAST VERTEX USED
C
      MAXVER=800L(PPDOHW(HUSED.WHEAD))      3 INTEGER POINTER IN REAL ARRAY!
      MSAOHW(HUSED.WHEAD)=MAXVER
C
C
C TRANSFORM VERTICES
C
      IF(MAXVER.GT.NVER+2) CALL MDWARN(
        - 'PRINT/PLOT POLYGRAM NOT AVAILABLE')
C      DO 200 NOD=NVER,MAXVER
C        CALL AXP(
C          &      ADJLIN,ADJSAM,
C          &      PPDOHW(WLIN,NOD)+PPDOHW(WLIN,WORIG),
C          &      PPDOHW(WCOL,NOD)+PPDOHW(WCOL,WORIG))
C          MSAOHW(WLIN,NOD)=IFIX(ADJLIN+.5)-MSAOHW(WLIN,WORIG)
C          MSAOHW(WSAM,NOD)=IFIX(ADJSAM+.5)-MSAOHW(WSAM,WORIG)
C 200 CONTINUE
C
      RETURN
      END

```



```

SUBROUTINE VER04U  & GEOGRAPHIC WINDOW VERTICES FROM UTM VERTICES
-----
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      10/07/73      ORIGINAL CODE
C      E H SCHLOSSER      LEC      10/30/79      UPGRADE DOCUMENTATION
C
C METHOD
C -----
C
C      TRANSFORM VERTICES IN THE OUTPUT UTM WINDOW PACKET INTO
C      VERTICES IN THE OUTPUT GEOGRAPHIC WINDOW PACKET.  ALL VERTEX
C      COORDINATES ARE RELATIVE TO THEIR RESPECTIVE WINDOW ORIGINS.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      G4U          & GEOGRAPHIC COORDINATES FOR UTM COORDINATES
C
C EXCEPTIONS
C -----
C
C      1. IF UTMCHD IS UNDEFINED, THEN THE RESULTS OF VER04U ARE UNDEFINED.
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMFIT.LIST      & COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C      INCLUDE KOMOWH.LIST      & COMMON OUTPUT WINDOW PACKETS
C      INCLUDE WINDEF.LIST      & DEFINE STRUCTURE OF WINDOW PACKETS
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER MAXVER          & MAXIMUM (LAST) VERTEX DEFINED (USED)
C      INTEGER NOD             & NODE NUMBER
C
C PROCEDURE
C -----
C
C      CALL TRACE
C

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

VER04U  
002

```
C
C INITIALIZE POINTER TO LAST VERTEX USED
C
      MAXVER=BOOL(UTMOWH(MUSED,MHEAD))      & INTEGER POINTER IN REAL ARRAY!
      GEDOWH(MUSED,MHEAD)=UTMOWH(MUSED,MHEAD)
C
C
C TRANSFORM VERTICES
C
      DO 200 NOD=MVER,MAXVER
        CALL Q4U(
          &      GEDOWH(WLAT,NOD),
          &      GEDOWH(WLON,NOD),
          -      UTMOWH(MEA,NOD)+UTMOWH(MEA,WORIG),
          &      UTMOWH(MNO,NOD)+UTMOWH(MNO,WORIG),
          &      UTMCHD)
          GEDOWH(WLAT,NOD)=GEDOWH(WLAT,NOD)-GEDOWH(WLAT,WORIG)
          GEDOWH(WLON,NOD)=GEDOWH(WLON,NOD)-GEDOWH(WLON,WORIG)
200 CONTINUE
C
      RETURN
      END
```

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

WARNB  
001

SUBROUTINE WARNB( 8 SUBMIT WARNING FOR MISSING/INVALID FIELD FROM UNIT 5  
1 HSWARN) 8 WARNING MESSAGE (STARTS WITH '\*' IF REQUIRED)  
-----

C  
C  
C  
C HISTORY  
C -----

E M SCHLOSSER	LEC	03/08/75	ORIGINAL CODE IN READS
E M SCHLOSSER	LEC	02/14/79	REVISE & MAKE SEPARATE SUBROUTINE
E M SCHLOSSER	LEC	09/27/79	APPEND INVALID FIELD TO WARNING MSG

C  
C  
C  
C METHOD  
C -----

IF FIELD IS MISSING AND NOT REQUIRED. FLAG REMAINING FIELDS AS NOT  
REQUIRED. OTHERWISE OUTPUT THE WARNING MESSAGE. FOLLOWED BY THE FIRST 23  
CHARACTERS OF THE INVALID FIELD. OR '(MISSING)'.

C  
C  
C  
C MACHINE-DEPENDENT CODE  
C -----

DIMENSION SPECIFICATIONS ASSUME 8 CHARACTERS PER SINGLE  
PRECISION INTEGER.

C  
C  
C  
C EXTERNAL REFERENCES  
C -----  
C

INTEGER ICE	8 INTEGER-CHARACTER-EQUIVALENT OF FIRST CHAR OF STRING
INTEGER LENCST	8 LENGTH OF CHARACTER STRING
MSWARN	8 SUBMIT WARNING DIAGNOSTIC
GETOKM	8 GET CHARACTER STRING DATA FIELD FROM BUFFER

C  
C  
C  
C EXCEPTIONS  
C -----

1. NONE.

C  
C  
C  
C GLOBAL DECLARATIONS  
C -----  
C

INCLUDE NULCHR.LIST	8 DEFINE NULL CHARACTER
INCLUDE KOMLUS.LIST	8 POINTERS/FLAGS/BUFFER FOR UNIT 5

C  
C  
C  
C LOCAL DECLARATIONS  
C -----  
C

INTEGER KNTMP(4)	8 FIRST 23 CHARS OF DATA FIELD + NULCHR
------------------	---

C

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

WARNB  
002

```

C
C PROCEDURE
C -----
C
C
C IF CURRENT UNIT 5 DATA FIELD IS MISSING & NOT REQUIRED. NO DIAGNOSTIC
C
      IF(LUSMAX.GT.0) GO TO 120      & DATA FIELD IS PRESENT
      IF(ICE(MSWARN).NE.ICE('')) LUSREQ=0
      IF(LUSREQ.EQ.0) GO TO 900      & MISSING FLD HAS OPTIONAL
C
C
C IF MESSAGE IS NON-NULL. APPEND CONTENTS OF DATA FIELD AND PASS TO MOWARN
C
      120 IF(LENCST(MSWARN.1).EQ.0) GO TO 900      & NO MESSAGE
      CALL MOV CST(KHTEMP,(1),(23), '(MISSING)',(1),(9), ' ')
      LUSLEN=0      & PREPARE TO GET CURRENT FIELD AGAIN
      CALL GETOKH(KHTEMP,(23),LUSLLM, LUSLLM,LUSIMO)
      CALL PUTCHR(KHTEMP,(24), NULCHR)
      CALL MOWARN( MSWARN,' --',' ',KHTEMP)
C
C
C DONE
C
      900 RETURN
      END

```

DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

MINEXT  
001

SUBROUTINE MINEXT( 3 COMPUTE INTERCEPT PAIRS FOR WINDOW EXTERIOR  
0 INCEPT. 3 INTERCEPT TABLE (MIN/MAX SAMPLE PAIRS)  
0 NCHAX. 3 NUMBER OF INTERCEPTS (ALWAYS EVEN)  
.  
1 MSALIN. 3 MSS LINE TO BE INTERCEPTED BY POLYGRAM  
1 MSASLO. 3 LOW ADJUSTED SAMPLE  
1 MSASHI) 3 HIGH ADJUSTED SAMPLE  
-----

C  
C  
C  
C HISTORY  
C -----  
C  
C E M SCHLOSSER LEC ORIGINAL CODE  
C J C CRISP LEC 11/28/79 ADD MSASLO & MSASHI ARGUMENTS  
C  
C  
C METHOD  
C -----  
C  
C THIS SUBROUTINE COMPUTES THE INTERCEPTS OF THE CURRENT MSS WINDOW  
C WITH A SPECIFIED SCAN LINE. THIS IS AN ADAPTATION OF THE FAST OCTANT ROTATION  
C POINT-IN-POLYGRAM ALGORITHM DEVELOPED BY SCHLOSSER. IT GIVES VALID RESULTS  
C FOR CLOCKWISE, COUNTER-CLOCKWISE, AND COMPOSITE POLYGRAMS, BOTH CONCAVE AND  
C CONVEX. ALL BOUNDARY POINTS ARE CONSIDERED INTERIOR. THE NUMBER OF WORDS  
C IN THE ARRAY INCEPT MUST BE GREATER THAN THE NUMBER OF VERTICES IN THE  
C POLYGRAM TO BE INTERCEPTED. IF THE POLYGRAM EXTENDS OUTSIDE THE ENVELOPE,  
C ONLY THAT PORTION WITHIN THE ENVELOPE IS USED.  
C  
C  
C MACHINE-DEPENDENT CODE  
C -----  
C  
C NONE  
C  
C  
C EXTERNAL REFERENCES  
C -----  
C  
C ISRTBA 3 INTEGER BUBBLE SORT ASCENDING  
C  
C  
C EXCEPTIONS  
C -----  
C  
C NONE  
C  
C  
C GLOBAL DECLARATIONS  
C -----  
C  
C  
C INCLUDE KOMOWM.LIST  
C INCLUDE WINDEF.LIST  
C INCLUDE ASHDEF.LIST  
C  
C

DAH PACKAGE APPENDIX N  
UTILITY ROUTINES

NINEXT  
000

C LOCAL DECLARATIONS

C -----

C

INTEGER INCEPT(1)        8 ARGUMENT

PARAMETER NODMIN=NVER+1

DEFINE IFXRND(REAL)=IFIX(REAL\*SIGN(.5,REAL))    8 ROUND EITHER POS OR NEG NO

C

C

C PROCEDURE

C -----

C

C

C INITIALIZE INTERCEPT GENERATION

C

NODMAX=MSAOWH(MUSED,MHEAD)

IF(NODMAX.LT.MVER+3) GO TO 800        8 USE ENVELOPE

MLINRO=MSALIN-MSAOWH(MLIN,MORIO)        8 RELATIVE TO ORIGIN

NP12=MIND(MAX0(MSAOWH(MLIN,MVER)-MLINRO,-1),+1)    8 -1.0.+1 PI

NCEP=2        8 INTERCEPT NUMBER (LEAVE ROOM FOR FIRST ENVELOPE INTERCEPT)

C

C

C COMPUTE POLYGRAM INTERCEPTS

C

DO 200 NOD=NODMIN,NODMAX

NP11=NP12

NP12=MIND(MAX0(MSAOWH(MLIN,NOD)-MLINRO,-1),+1)    8 -1.0.+1 PI

IF((NP12-NP11).EQ.0) GO TO 200        8 NO INTERCEPT

INCEPT(NCEP)=(NP12-NP11)\*3        8 INCREMENTAL ROTATION (BIASED POSITIVE)

ASHM1(INCEPT(NCEP))=IFXRND(

8 FLOAT(MSAOWH(MSAM,MORIO)+MSAOWH(MSAM,NOD-1))+

8 FLOAT((MLINRO                    -MSAOWH(MLIN,NOD-1))+

8 (MSAOWH(MSAM,NOD)-MSAOWH(MSAM,NOD-1)))/

8 FLOAT(MSAOWH(MLIN,NOD)-MSAOWH(MLIN,NOD-1)))

NCEP=NCEP+1

200 CONTINUE

C

C

C COMPUTE ENVELOPE INTERCEPTS

C

INCEPT(1)=+3        8 NULL ROTATION (BIASED POSITIVE)

ASHM1(INCEPT(1))=MSASLO-1    8 SHIFT OUTSIDE ENVELOPE

INCEPT(NCEP)=+9999        8 FORCE TERMINATION

ASHM1(INCEPT(NCEP))=MSASHI+1    8 SHIFT OUTSIDE ENVELOPE

C

C

C SORT INTERCEPTS BY SAMPLE NUMBER

C

CALL ISRTBA(INCEPT,NCEP)

C

C

C INITIALIZE INTERCEPT REDUCTION

C

NPISUM=0        8 CUMULATIVE ROTATION

NCIN=1

NCOUT=1

GO TO 310

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

NINEXT  
888

```

C
C
C FIND STARTING INTERCEPT OF PAIR (FIRST OUTSIDE POLYORAM)
C
300 NCIN=NCIN+1
    IF(NCIN.GE.NCEP) GO TO 400
310 NPISUM=NPISUM+ASHM2(INCEPT(NCIN))-3      & CUMULATIVE ROTATION
    IF(NPISUM.NE.0) GO TO 300      & NOT OUTSIDE POLYORAM
330 INCEPT(NCOUT)=INCEPT(NCIN)/2+.18+1    & UNPACK, EXTEND SIGN & SHIFT OUTSIDE
C
C
C FIND ENDING INTERCEPT OF PAIR (LAST OUTSIDE POLYORAM)
C
    NCIN=NCIN+1
    NPISUM=NPISUM+ASHM2(INCEPT(NCIN))-3      & CUMULATIVE ROTATION
    IF(NPISUM.EQ.0) GO TO 330      & STILL OUTSIDE POLYORAM
    INCEPT(NCOUT+1)=INCEPT(NCIN)/2+.18+1  & UNPACK, EXTEND SIGN, SHIFT OUTSIDE
    IF(INCEPT(NCOUT+1).LT.INCEPT(NCOUT)) GO TO 300  & DELETE TRIVIAL PAIR
    NCOUT=NCOUT+2
    GO TO 300
C
C
C STORE NUMBER OF INTERCEPTS
C
400 NCMAX=NCOUT-1
    GO TO 900
C
C
C DEGENERATE POLYORAM -- USE ENVELOPE
C
800 NCMAX=0
C
C
C NORMAL RETURN
C
900 RETURN
    END

```

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

MININT  
001

```

SUBROUTINE MININT( 8 COMPUTE INTERCEPTS FOR WINDOW INTERIOR
O INCEPT, 8 INTERCEPT TABLE (MIN/MAX SAMPLE PAIRS)
O NCMAX, 8 NUMBER OF INTERCEPTS (ALWAYS EVEN)
)
I MSALIN) 8 MSS LINE TO BE INTERCEPTED BY POLYGRAM
-----
C
C
C (E M SCHLOSSER) ***** NOT YET COMPLETED *****
C
C
C THIS SUBROUTINE COMPUTES THE INTERCEPTS OF THE CURRENT MSS WINDOW
C WITH A SPECIFIED SCAN LINE. THIS IS AN ADAPTATION OF THE FAST OCTANT ROTATION
C POINT-IN-POLYGRAM ALGORITHM DEVELOPED BY SCHLOSSER. IT GIVES VALID RESULTS
C FOR CLOCKWISE, COUNTER-CLOCKWISE, AND COMPOSITE POLYGRAMS, BOTH CONCAVE AND
C CONVEX. ALL BOUNDARY POINTS ARE CONSIDERED INTERIOR. THE NUMBER OF WORDS
C IN THE ARRAY INCEPT MUST NOT BE LESS THAN THE NUMBER OF VERTICES IN THE
C POLYGRAM TO BE INTERCEPTED.
C
C
C EXTERNAL SUBROUTINES/FUNCTIONS CALLED
C -----
C
C
C ISRTBA
C
C
C INCLUDE KOMOWH.LIST
C INCLUDE MINDEF.LIST
C INCLUDE ASMDEF.LIST
C DIMENSION INCEPT(1)
C PARAMETER NOCMIN=NCMAX+1
C
C
C INITIALIZE INTERCEPT GENERATION
C
C
C NOCMAX=MSAOWH(MUSED,MHEAD)
C IF(NOCMAX.LT.MVER+3) GO TO 000 8 USE ENVELOPE
C MLINRO=MSALIN-MSAOWH(MLIN,MORIO) 8 RELATIVE TO ORIGIN
C NP12=MING(MAX0(MSAOWH(MLIN,MVER)-MLINRO,-1),+1) 8 -1.0,+1 PI
C NCEP=1 8 INTERCEPT NUMBER
C
C
C COMPUTE POLYGRAM INTERCEPTS
C
C
C DO 200 NOD=NOCMIN,NOCMAX
C NP11=NP12
C NP12=MING(MAX0(MSAOWH(MLIN,NOD)-MLINRO,-1),+1) 8 -1.0,+1 PI
C INCEPT(INCEP)=NP12-NP11 8 INCREMENTAL ROTATION IN M2
C IF(INCEPT(INCEP).EQ.0) GO TO 200 8 NO INTERCEPT
C INCEPT(INCEP)=ABS(INCEPT(INCEP)+2) 8 FORCE POSITIVE OR PLUS ZERO
C ASHNI(INCEPT(INCEP))-IFIX(
C 8 FLOAT(MSAOWH(MSAN,MORIO)+MSAOWH(MSAN,NOD-1))
C 8 FLOAT(MLINRO-MSAOWH(MLIN,NOD-1))
C 8 (MSAOWH(MSAN,NOD)-MSAOWH(MSAN,NOD-1))/
C 8 FLOAT(MSAOWH(MLIN,NOD)-MSAOWH(MLIN,NOD-1))
C 8 0.5)
C NCEP=NCEP+1

```



DAM PACKAGE APPENDIX N  
UTILITY ROUTINES

MININT  
002

```

200 CONTINUE
C
C
C SORT INTERCEPTS BY SAMPLE NUMBER
C
    NCEP=NCEP-1
    IF(NCEP.NE.0) GO TO 250
    NCMAX=0
    GO TO 300
250 CALL ISRTDA(INCEPT,NCEP)
C
C
C INITIALIZE INTERCEPT REDUCTION
C
    NPISUM=0      & CUMULATIVE ROTATION
    NCIN=1
    NCOUT=1
    GO TO 310
C
C
C FIND STARTING INTERCEPT OF PAIR
C
300 NCIN=NCIN+1
310 NPISUM=NPISUM+ASHM2(INCEPT(NCIN))-2      & REMOVE BIAS
    INCEPT(NCOUT)=INCEPT(NCIN)/2+10      & UNPACK & EXTEND SIGN
    NCOUT=NCOUT+1
C
C
C FIND ENDING INTERCEPT OF PAIR
C
400 NCIN=NCIN+1
    NPISUM=NPISUM+ASHM2(INCEPT(NCIN))-2      & REMOVE BIAS
    IF(NPISUM.NE.0) GO TO 400
    INCEPT(NCOUT)=INCEPT(NCIN)/2+10      & UNPACK & EXTEND SIGN
    NCOUT=NCOUT+1
    IF(NCIN.LT.NCEP) GO TO 300
C
C
C STORE NUMBER OF INTERCEPTS
C
    NCMAX=NCOUT-1
    GO TO 300
C
C
C DEGENERATE POLYGRAM -- USE ENVELOPE
C
500 INCEPT(1)=MSAOWM(MSAH,MMIN)
    INCEPT(2)=MSAOWM(MSAH,MMAX)
    NCMAX=2
C
C
C NORMAL RETURN
C
600 RETURN
END

```

**DAN PACKAGE APPENDIX N  
UTILITY ROUTINES**

**WRITE4  
001**

```

SUBROUTINE WRITE4( 8 WRITE TO UNIT 4 (SDF COMMAND RECALL FILE)
*
I NUMCRD. 8 CARD IMAGE NUMBER
I IMAGE1 8 IMAGE (14 WORDS, FIELDATA)
-----
C
C
C
C      E H SCHLOSSER      LEC      03/17/78      ORIGINAL CODE
C      E H SCHLOSSER      LEC      11/07/78      WRITE EOF AFTER EACH NON-BATCH IMAGE
C
C
C METHOD
C -----
C
C      OPEN:  ASSIGN TEMPORARY RECALL FILE.
C              WRITE UNIVAC EXEC-8 SDF LABEL IMAGE.
C      WRITE:  TEMPORARILY INSERT UNIVAC EXEC-8 SDF CONTROL WORDS BEFORE 8
C              AFTER IMAGE.
C              WRITE IMAGE INCLUDING CONTROL WORDS TO FASTRAND SECTOR 8 NUMCRD.
C              IF NOT BATCH WRITE UNIVAC EXEC-8 SDF EOF CONTROL WD TO NEXT SECTOR.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      VERRRY!!
C
C
C EXTERNAL REFERENCES
C -----
C
C      MOWARN      8 PRINT/LOG/COUNT 'WARNING' DIAGNOSTIC MESSAGE
C      ERION       8 INITIATE I/O AND WAIT FOR COMPLETION
C      MYCONT      8 MOVE CONTENTS BETWEEN TWO MACHINE ADDRESSES
C      ERCSF       8 SUBMIT UNIVAC EXEC-8 FACILITY REQUEST CONTROL STATEMENT
C
C
C EXCEPTIONS
C -----
C
C      1. IF NUMCRD < 1. THE IMAGE IS NOT WRITTEN TO THE RECALL FILE.
C
C      2. IF NUMCRD > 9000. A WARNING DIAGNOSTIC IS GENERATED.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMKOT.LIST      8 COMMON PROGRAM EXECUTION SWITCHES, COUNTERS
C      INCLUDE KOMIO.LIST      8 COMMON I/O FUNCTIONS
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER IMAGE(14)      8 ARGUMENT
C      INTEGER ITEMP00(8)     8 TEMPORARY STORAGE FOR IMAGE(8)

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

WRITE4  
002

```

      INTEGER ITHPIS          & TEMPORARY STORAGE FOR IMAGE(15)
      INTEGER LU4PKT(6)      & I/O PACKET
      DATA (LU4PKT(1),1-1,2) /'RECALL'      & FILE NAME
      INTEGER KWD14/0001600000000/ & SDF CONTROL WD: 14 WD DATA IMAGE FOLLOWS
      INTEGER KWB12/0401400000000/ & SDF CONTROL WD: 12 WD BYPASS IMAGE FOLLOWS
      INTEGER KWE0F /077777777777/ & SDF CONTROL WD: END-OF-FILE

C
C
C PROCEDURE
C -----
C
C
C CHECK NUMCRD
C
      IF(NUMCRD.LE.0) GO TO 900
      IF(NUMCRD.GT.5000) CALL MDHARN('RECALL FILE OVERFLOW')
C
C
C SAVE CONTENTS OF LOCATIONS USED FOR SDF CONTROL WORDS
C
      CALL MVCONT(LOC(IMAGE)-1,LOC(ITHP00))      & ITHP00=IMAGE(0)
      ITHP15=IMAGE(15)
C
C
C SET UP SDF CONTROL WORDS (1 CARD IMAGE PER 28 WORD SECTOR)
C
      CALL MVCONT(LOC(KWD14),LOC(IMAGE)-1)      & IMAGE(0)=KWD14
      IMAGE(15)=KWB12      & 12 WORD BYPASS IMAGE FOLLOWS
C
C
C INITIALIZE PACKET & WRITE IMAGE TO DISK SECTOR
C
      IOSIZE(LU4PKT)=16
      IOADDR(LU4PKT)=LOC(IMAGE)-1      & LOC(IMAGE(0))
      IOSECT(LU4PKT)=NUMCRD
      IOFUNC(LU4PKT)='3C'      & WRITE
      CALL ERION(LU4PKT)
C
C
C RESTORE CONTENTS OF LOCATIONS USED FOR SDF CONTROL WORDS
C
      CALL MVCONT(LOC(ITHP00),LOC(IMAGE)-1)      & IMAGE(0)=ITHP00
      IMAGE(15)=ITHP15
C
C
C IF NOT BATCH, WRITE SDF END-OF-FILE CONTROL WORD TO NEXT SECTOR
C
      IF(MBATCH.NE.0) GO TO 900      & BATCH
      IOSIZE(LU4PKT)=1
      IOADDR(LU4PKT)=LOC(KWE0F)
      IOSECT(LU4PKT)=NUMCRD+1
      IOFUNC(LU4PKT)='3C'      & WRITE
      CALL ERION(LU4PKT)
      GO TO 900
C
C

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

WRITE4  
003

```

C
C
C   ENTRY OPEN4  & OPEN UNIT 4 (SDF COMMAND RECALL FILE)
C   -----
C
C   INTEGER KHLABL/0503313000000/  & SDF CONTROL NO: LABEL IMAGE
C
C   CALL TRACE('OPEN4')
C
C   ASSIGN TEMPORARY RECALL FILE
C
C
C   CALL ERCSF(NA0,'BASQ.T RECALL. . ')
C
C
C   WRITE SDF LABEL IMAGE IN 0TH SECTOR
C
C   IOSIZE(LU4PKT)=1
C   IOADDR(LU4PKT)=LOC(KHLABL)
C   IOSECT(LU4PKT)=0
C   IOFUNC(LU4PKT)='&C'  & WRITE
C   CALL ERIOH(LU4PKT)
C
C
C   COMMON RETURN
C
C   900 RETURN
C   END

```

DAN PACKAGE APPENDIX N  
UTILITY ROUTINES

NRVERT  
001

```

      SUBROUTINE NRVERT( 8 WRITE WINDOW VERTEX COORDINATES
      I WINDOW.          8 WINDOW PACKET (REAL)
      I DIV1,DIV2.        8 COORDINATE DIVISORS (FOR SCALING OUTPUT)
      I MATARG.           8 FORMAT SPECIFICATION
      I NODMAX)           8 NODE CONTAINING LAST VERTEX TO BE WRITTEN
      -----
C
C
C (E H SCHLOSSER)
C
C
      DIMENSION MATARG(1),MATFIX(10)      8 FIX FOR UNIVAC FORTRAN V BUG
      DIMENSION WINDOW(2,1),RLTEMP(2)
      INCLUDE WINDEF.LIST
      PARAMETER NODMIN=NRVER+1
C
C
C FIX FOR UNIVAC FORTRAN V BUG -- INCORRECT CODE GENERATED WHEN ARRAY
C CONTAINING FORMAT SPECIFICATION IS PASSED AS ARGUMENT
C
      DO 100 N=1,10
      IF(MATARG(N).EQ.-0) GO TO 150      8 LITERAL STOP WD IS -0
100 MATFIX(N)=MATARG(N)
C
C
C WRITE VERTEX COORDINATES
C
150 JCOMMA='.'
      DO 200 N=NODMIN,NODMAX
      RLTEMP(1)=WINDOW(1,N)/DIV1
      RLTEMP(2)=WINDOW(2,N)/DIV2
      IF(N.EQ.NODMAX) JCOMMA=' '
      WRITE(8,MATFIX) RLTEMP,JCOMMA
200 CONTINUE
      RETURN
      END

```

DAM PACKAGE APPENDIX H  
UTILITY ROUTINES

XREG77  
891

```

SUBROUTINE XREG77( 8 DUMP SPECIFIED UNIVAC 1100 X-REGISTER IN OCTAL
1 NUMREG) 8 NUMBER OF X-REGISTER (0 THRU 11)
-----

HISTORY
-----

E H SCHLOSSER      LEC      12/05/79      ORIGINAL CODE

METHOD
-----

ENCODE CONTENTS OF UNIVAC 1100 SERIES X-REGISTER IN OCTAL AND PRINT THEM.

THIS ROUTINE IS DESIGNED TO HELP DEBUG OBSCURE ERRORS IN THE
COMPILER & OPERATING SYSTEM -- ON RETURN, ALL REGISTERS (EVEN THOSE IN
THE VOLATILE 'MINOR REGISTER SET') ARE UNCHANGED.

MACHINE-DEPENDENT CODE
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 8-BIT
FIELDATA CHARACTERS. THE CALLING SEQUENCE IS THAT USED BY UNIVAC
FORTRAN V.
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.
DIFFERENT COMPILERS (EG., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

EXTERNAL REFERENCES
-----

ER PRINTS      8 UNIVAC X-8 EXEC REQUEST TO PRINT FIELDATA CHARACTER BUFFER

EXCEPTIONS
-----

1. THE ROUTINE PRINTS A DIAGNOSTIC IF 0 <= NUMREG <= 11.

GLOBAL DECLARATIONS
-----

NONE.

AXRS      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
S(00) . 1-BANK
XREG77:    DS      A0,ABUF+0      .
           DS      A2,ABUF+2      .
           DS      A4,ABUF+4      .
           SX      X0,XBUF+0      .

```

DAN PACKAGE APPENDIX M  
UTILITY ROUTINES

XRE077  
002

	SX	X1.XBUF+1	.
	SX	X2.XBUF+2	.
	SX	X3.XBUF+3	.
	SX	X4.XBUF+4	.
	SX	X5.XBUF+5	.
	SX	X6.XBUF+6	.
	SX	X7.XBUF+7	.
	SX	X8.XBUF+8	.
	SX	X9.XBUF+9	.
	SX	X10.XBUF+10	.
	SX	X11.XBUF+11	.
.			
SUBNAME	LA	A3.1.X11	. A3 := M.B. WORD (M2 = ADDR OF SUBR NAME)
	LA	A1.0.A3	. A1 := SUBROUTINE NAME
	SA	A1.PBUF+0	. PUT SUBROUTINE NAME IN PRINT BUFFER
.			
REONUM	LA	A0.0.X11	. A0 := NUMREG
	JN	A0.BADNUM	. BAD IF NUMREG NEGATIVE
	TG.U	A0.12	. SKIP NI IF 12 > NUMREG
	J	BADNUM	. (NUMREG >= 12)
.			
	LA	A1.BIAS	. '0' IS NEGATIVE!
	LA	A2.BIAS	.
	LA	A3.XBUF.A0	. A3 := CONTENTS OF SPECIFIED X-REG
.			
UNPACK8	LOSL	A1.6	. SHIFT & PAD
	SSL	A2.3	. TOO FAR!
	LOSL	A2.3	. EXTRACT 3-BIT OCTAL DIGIT
	JN	A1.UNPACK8	. REPEAT TIL LEADING CHAR NOT '0'
.			
ENCODE8	AA	A1.BIAS	. CONVERT 8-BIT OCTAL
	AA	A2.BIAS	. TO FIELDATA
	DS	A1.PBUF+3	. AND PUT IN PRINT BUFFER
.			
ENCODE10	DSL	A0.38	. A1 := A0, A0 := 0
	DI.U	A0.10	. A0 := TENS DIGIT, A1 := UNITS DIGIT
	AA.U	A0.'888880'	. CONVERT TENS DIGIT TO FIELDATA CHAR
	AA.U	A1.'888880'	. CONVERT UNITS DIGIT TO FIELDATA CHAR
	SA.S2	A0.PBUF+2	. INSERT TENS CHARACTER IN BUFFER
	SA.S3	A1.PBUF+2	. INSERT UNITS CHARACTER IN BUFFER
.			
PRINT	PSRINT	(PF 1.5.PBUF)	. PRINT BUFFER
.			
RESTORE	OL	A0.ABUF+0	. RESTORE REGISTERS
	OL	A2.ABUF+2	.
	OL	A4.ABUF+4	.
	J	2.X11	. RETURN
.			
BADNUM	OL	A1.BADMS0	. PUT BAD MS0 INTO ...
	DS	A1.PBUF+3	. ... PRINT BUFFER
	J	ENCODE10	. ENCODE BAD NUMREG & PRINT BUFFER
.			
S(01) . D-BANK			
ABUF	RES	0	.
XBUF	RES	16	.
PBUF	RES	1	. 1 WD FOR SUBROUTINE NAME

**DAM PACKAGE APPENDIX N  
UTILITY ROUTINES**

**XRE077  
003**

		• XRE0 99 •	
	RES	2	• 2 WDS FOR OCTAL ENCODING OF REGISTER
BADMSO		• (BAD NUMBER) •	
PF	FORM	12.8.18	
BIAS		• 000000 •	
	END		



**PREFACE TO APPENDIX O**  
-----

**THE ROUTINES IN THIS APPENDIX TRANSFORM COORDINATES AMONG THE FOLLOWING  
COORDINATE SYSTEMS:**

- A** ADJUSTED MSS COORDINATES (LINE, SAMPLE)  
(AS DATA IS STORED ON 'X', 'AM', OR 'PM' COMPUTER TAPE)
- G** GEOGRAPHIC COORDINATES (DEGREES NORTH, DEGREES WEST)
- P** PRINT/PLOT DEVICE COORDINATES (LINE, COLUMN)
- U** TRANSVERSE MERCATOR COORDINATES (METERS EAST, METERS NORTH)  
(USED WITH UTM ZONE CENTRAL MERIDIAN FOR UTM)  
(USED WITH SCENE CENTER CENTRAL MERIDIAN FOR STM)

**FOR THE FUNCTIONS USED TO MODEL THE ORBITAL AND SCANNER GEOMETRY. SEE  
TRFORM-PROCS IN APPENDIX Q.**

**DAM PACKAGE APPENDIX 0  
COORDINATE TRANSFORMATIONS**

**APPENDIX-0  
001**

<b>SPRT.SC DAM.PREFACE-0</b>	<b>. (0000)</b>	<b>SET TABS 8 31</b>
<b>SPRT.SC DAM.APPENDIX-0</b>	<b>.</b>	
<b>SPRT.SC DAM.A40</b>	<b>.</b>	<b>ADJUSTED MSS COORDINATES FOR GEOGRAPHIC COORD</b>
<b>SPRT.SC DAM.A4P</b>	<b>.</b>	<b>ADJUSTED MSS COORDINATES FOR PRT/PLT COORDINATES</b>
<b>SPRT.SC DAM.DU40/CLARKE1886</b>	<b>.</b>	<b>DBL PRECISION UTM COORDINATES FOR GEOGRAPHIC</b>
<b>SPRT.SC DAM.04A</b>	<b>.</b>	<b>GEOGRAPHIC COORDINATES FOR ADJUSTED MSS COORD</b>
<b>SPRT.SC DAM.04P</b>	<b>.</b>	<b>GEOGRAPHIC COORDINATES FOR PRT/PLT COORD</b>
<b>SPRT.SC DAM.04U/CLARKE1886</b>	<b>.</b>	<b>GEOGRAPHIC COORDINATES FOR UTM COORDINATES</b>
<b>SPRT.SC DAM.P4A</b>	<b>.</b>	<b>PRT/PLT COORDINATES FOR ADJUSTED MSS COORDINATES</b>
<b>SPRT.SC DAM.P4G</b>	<b>.</b>	<b>PRT/PLT COORDINATES FOR GEOGRAPHIC COORDINATES</b>
<b>SPRT.SC DAM.U40/CLARKE1886</b>	<b>.</b>	<b>UTM COORDINATES FOR GEOGRAPHIC COORDINATES</b>

DAM PACKAGE APPENDIX O  
COORDINATE TRANSFORMATIONS

A40  
001

SUBROUTINE A40(    : ADJUSTED MSS COORDINATES FOR GEOGRAPHIC COORDINATES  
O ADJLIN,ADJSAM.   : SCANNER LINE, SAMPLE (LINE-LENGTH ADJUSTED)  
I GEOLAT,GEOLON)   : LATITUDE, LONGITUDE (DEGREES)  
-----

```

C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      05/23/73      ORIGINAL CODE IN 02A
C      E M SCHLOSSER      LEC      10/29/79      RENAME A40 & REVISE ARG SEQUENCE
C
C METHOD
C -----
C
C      TRANSFORM GEOGRAPHIC COORDINATES TO SCENE TRANSVERSE MERCATOR COORDINATES.
C      TRANSFORM SCENE TRANSVERSE MERCATOR COORDINATES TO CORRECTED COORDINATES.
C      TRANSFORM CORRECTED COORDINATES TO ADJUSTED SCANNER COORDINATES.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      U40               : UTM (OR STM) COORDINATES FROM GEOGRAPHIC COORDINATES
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMNER.LIST       : COMMON ERTS SCENE PARAMETERS
C      INCLUDE KONFIT.LIST       : COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C      INCLUDE TRFORM.LIST       : DEFINE COORDINATE TRANSFORMATION FUNCTIONS
C
C LOCAL DECLARATIONS
C -----
C
C      DEFINE CORLIN=CORL4S(STHE,STMN)       : CORRECTED LINE
C      DEFINE CORSAM=CORS4S(STHE,STMN)       : CORRECTED SAMPLE
C
C PROCEDURE
C -----
C
C      CALL U40(STHE,STMN,   GEOLAT,GEOLON,STMCHD)
C      ADJLIN=ADJL4C(CORLIN,CORSAM)       : CORLIN, CORSAM DEFINED ABOVE
C      ADJSAM=ADJS4C(CORLIN,CORSAM)
C

```

**DAN PACKAGE APPENDIX 0  
COORDINATE TRANSFORMATIONS**

**A40  
002**

**RETURN  
END**

DAN PACKAGE APPENDIX O  
COORDINATE TRANSFORMATIONS

A4P  
001

```

SUBROUTINE A4P( 8 ADJUSTED MSS COORD FOR PRINT/PLOT COORD
0 ADJLIN. 8 ADJUSTED SCAN LINE
0 ADJSAM. 8 ADJUSTED SCAN SAMPLE
-
1 PPOLIN. 8 PRINT/PLOT LINE
1 PPOCOL) 8 PRINT/PLOT COLUMN
-----
C
C
C HISTORY
C -----
C      E H SCHLOSSER      LEC      05/24/73      ORIGINAL CODING (P2A)
C      D A BECK          LEC      10/23/79      RENAME/REVISE A4P
C
C METHOD
C -----
C
C      TRANSFORM PRINT/PLOT TO 'CORRECTED' COORDINATES.
C      IF OUTPUT IS SCALED THEN
C          CORRECT FOR ORBIT AND SCAN GEOMETRY
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      NONE.
C
C EXCEPTIONS
C -----
C
C      1. IF PPD TRANSFORMATIONS ARE 'SPACED' AND CALSPA HAS NOT
C          BEEN CALLED THEN RESULTS ARE UNDEFINED.
C
C      2. IF PPD TRANSFORMATIONS ARE 'SCALED' AND CALSCA HAS NOT
C          BEEN CALLED THEN RESULTS ARE UNDEFINED.
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KONNER.LIST      8 COMMON ERTS SCENE PARAMETERS
C      INCLUDE KONFIT.LIST     8 COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C      INCLUDE TRFORM.LIST     8 DEFINE COORDINATE TRANSFORMATIONS
C
C LOCAL DECLARATIONS
C -----

```

**BAH PACKAGE APPENDIX 0  
COORDINATE TRANSFORMATIONS**

**A4P  
002**

```
C
      REAL CORLIN          S CORRECTED SCANNER LINE
      REAL CORSAM         S CORRECTED SCANNER SAMPLE
C
C
C PROCEDURE
C -----
C
C
C TRANSFORM PRINT/PLOT TO 'CORRECTED' COORDINATES
C
      CORLIN=CORL4P(PPDLIN,PPDCOL)
      CORSAM=CORS4P(PPDLIN,PPDCOL)
C
C
C IF REPRESENTATIVE FRACTION IS DEFINED, DO SCALING ELSE DO SPACING
C
      IF(IIRFD.EQ.0) GO TO 200
C
C
C 'SCALED' OUTPUT -- CORRECTED FOR ORBIT AND SCAN GEOMETRY
C
      ADJLIN=ADJL4C(CORLIN,CORSAM)
      ADJSAM=ADJS4C(CORLIN,CORSAM)
      GO TO 900
C
C
C 'SPACED' OUTPUT -- NOT CORRECTED FOR ORBIT AND SCAN GEOMETRY
C
200 ADJLIN=CORLIN
      ADJSAM=CORSAM
C
C
C RETURN TO CALLING ROUTINE
C
900 RETURN
      END
```

**DAN PACKAGE APPENDIX D  
COORDINATE TRANSFORMATIONS**

**DU40/CLARKE1888  
001**

SUBROUTINE DU40: 0 UTM COORDINATES FOR GEOGRAPHIC COORDINATES (DBLE PREC  
0 UTME,UTMN. 0 UNIVERSAL TRANSVERSE MERCATOR EASTING, NORTHING (METRES)  
1 GEOLAT,GEOLON. 0 GEOGRAPHIC (DEGREES) LATITUDE, LONGITUDE  
1 CHDGE) 0 PROJECTION CENTRAL MERIDIAN (DEGREES)  
-----

C  
C  
C  
C HISTORY  
C -----

C E H SCHLOSSER LEC 02/13/73 ORIGINAL CODE IN D02U  
C E H SCHLOSSER LEC 03/20/79 RENAME U40 & REVISE ARG SEQUENCE

C  
C  
C  
C METHOD  
C -----

C THIS SUBROUTINE TRANSFORMS GEOGRAPHIC COORDINATES IN DEGREES TO UTM  
C (UNIVERSAL TRANSVERSE MERCATOR) COORDINATES IN METERS, RELATIVE TO THE  
C SPECIFIED CENTRAL MERIDIAN. ALL COMPUTATIONS ARE ON THE CLARKE 1888  
C SPHEROID, WHICH IS STANDARD FOR ALL OF NORTH AMERICA. LATITUDE IS ASSUMED  
C NORTH, AND MUST BE POSITIVE. LONGITUDE IS ASSUMED WEST AND MUST BE  
C POSITIVE. THE USER MAY SPECIFY EITHER A STANDARD (103-0-ZONE) OR A  
C NON-STANDARD CENTRAL MERIDIAN.

C  
C  
C MACHINE-DEPENDENT CODE  
C -----

C MAXIMUM ERROR IN THE TRANSFORMED COORDINATES IS 0.001 METER FOR POINTS  
C WITH LATITUDES BETWEEN 0 AND 90 DEGREES AND LONGITUDES WITHIN 5 DEGREES  
C OF THE CENTRAL MERIDIAN. (THIS ASSUMES ALL COMPUTATIONS IN DOUBLE  
C PRECISION ON A COMPUTER WITH A WORD SIZE OF AT LEAST 32 BITS.)

C THIS SOURCE CODE IS DESIGNED FOR A COMPILER THAT GENERATES SUBROUTINE  
C CODE WHICH PRESERVES THE VALUES OF LOCAL VARIABLES BETWEEN SUCCESSIVE  
C CALLS. WHEN A LARGE NUMBER OF POINTS ON A REGULAR GRATICULE ARE TO BE  
C TRANSFORMED, GREATER EFFICIENCY WILL RESULT IF THEY ARE GROUPED BY  
C LATITUDE.

C THIS SOURCE CODE IS DESIGNED TO TAKE ADVANTAGE OF THE OPTIMIZATION  
C CAPABILITY OF THE UNIVAC FORTRAN V COMPILER, WHICH RECOGNIZES COMMON  
C SUB-EXPRESSIONS AND GENERATES CODE TO PRE-COMPUTE AND STORE THEIR VALUES  
C IN AVAILABLE REGISTERS. EXPLICIT SOURCE CODE OPTIMIZATION MAY BE  
C APPROPRIATE FOR OTHER COMPILERS.

C  
C  
C EXTERNAL REFERENCES  
C -----

C NONE (OTHER THAN SYSTEM ROUTINES).

C  
C  
C GLOBAL DECLARATIONS  
C -----

**DAN PACKAGE APPENDIX C  
COORDINATE TRANSFORMATIONS**

**BUM/CLARKE1886  
002**

```

C
C      NONE.
C
C
C
C LOCAL DECLARATIONS
C -----
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION FE      / .50*8/      8 CENT MERIDIAN FALSE EASTING = 500000 M
      DOUBLE PRECISION OLDLAT/.50*8/      8 OLD LATITUDE (INITIALLY IMPOSSIBLE)
      DEFINE SECLAT=OECLAT*.360*4      8 LATITUDE IN SECONDS
      DEFINE RADLAT=OECLAT*.314159265359*1/.180*3      8 LATITUDE IN RADIANS
      DEFINE P=(CHDEG-OECLON)*.360*8
      DOUBLE PRECISION COS01      8 COSINE**1
      DOUBLE PRECISION COS02      8 COSINE**2
      DOUBLE PRECISION SINCOS      8 SIN**COS
C
C
C PROCEDURE
C -----
C
C CALL DCOS & DSQRT ONLY IF LATITUDE IS DIFFERENT FROM LAST POINT
C
      IF(OLDLAT.EQ.OECLAT) GO TO 200
      OLDLAT=OECLAT
      COS01=DCOS(RADLAT)
      COS02=DCOS(RADLAT)**2
      SINCOS=DSQRT(1.-DCOS(RADLAT)**2)*DCOS(RADLAT)
      ECCFAC=1./DSQRT(1.+ .681478490-2*DCOS(RADLAT)**2)
200 CONTINUE
C
C
C COMPUTE EASTING
C
      UTM-E=FE
      8 *P      8 .31019182110*8*COS01*ECCFAC
      8      8 *1.
      8 *P**2      8 .3817405080-3
      8      8 *(1.-.20*1*COS02-.6814780-2*COS02**2)
      8 *P**4      8 .4803820-7
      8      8 *(1.-.20*2*COS02+.2360470*2*COS02**2
      8      8      8 +.48070*0*COS02**3)
C
C
C COMPUTE NORTHING
C
      UTMN=.3089787881410*2*(SECLAT
      8 -1.18528938430*4-.44833860*1*COS02+.235580-1*COS02**2)
      8      8 *SINCOS)
      8 *P**2      8 .75182847280*4*SINCOS*ECCFAC
      8      8 *1.
      8 *P**2      8 .18587830-3
      8      8 *(-1.-.60*1*COS02+.61333880-1*COS02**2
      8      8      8 +.185770-3*COS02**3)
      8 *P**4      8 .183480-7
      8      8 *(1.-.60*2*COS02+.117750*3*COS02**2

```



DAN PACKAGE APPENDIX O  
COORDINATE TRANSFORMATIONS

DU40/CLARKE1888  
003

8  
RETURN  
END

♦.40890♦1♦C0502♦♦311

DAN PACKAGE APPENDIX O  
COORDINATE TRANSFORMATIONS

04A  
001

SUBROUTINE 04A(   8 GEOGRAPHIC COORDINATES FOR ADJUSTED MSS COORDINATES  
0 GEOLAT,GEOLON.   8 LATITUDE, LONGITUDE (DEGREES)

1 ADJLIN,ADJSAM)   8 SCANNER LINE, SAMPLE (LINE-LENGTH ADJUSTED)  
-----

C  
C  
C  
C HISTORY  
C -----

C       E M SCHLOSSER       LEC       09/29/73       ORIGINAL CODE IN A20  
C       E M SCHLOSSER       LEC       10/29/79       RENAME 04A & REVISE ARG SEQUENCE

C  
C  
C  
C METHOD  
C -----

C       TRANSFORM ADJUSTED SCANNER COORDINATES TO CORRECTED COORDINATES.  
C       TRANSFORM CORRECTED COORDINATES TO SCENE TRANSVERSE MERCATOR COORDINATES.  
C       TRANSFORM SCENE TRANSVERSE MERCATOR COORDINATES TO GEOGRAPHIC COORDINATES.

C  
C  
C  
C MACHINE-DEPENDENT CODE  
C -----

C       NONE.

C  
C  
C  
C EXTERNAL REFERENCES  
C -----

C       04U               8 GEOGRAPHIC COORDINATES FROM UTM (OR STM) COORDINATES

C  
C  
C  
C GLOBAL DECLARATIONS  
C -----

C       INCLUDE KOMNER.LIST       8 COMMON ERTS SCENE PARAMETERS  
C       INCLUDE KOMFIT.LIST       8 COMMON ADJUSTMENT/REGISTRATION PARAMETERS  
C       INCLUDE TRFORM.LIST       8 DEFINE COORDINATE TRANSFORMATION FUNCTIONS

C  
C  
C  
C LOCAL DECLARATIONS  
C -----

C       DEFINE CORLIN=CORL4(ADJLIN,ADJSAM)  
C       DEFINE CORSAM=CORS4(ADJLIN,ADJSAM)

C  
C  
C  
C PROCEDURE  
C -----

C       STMN=STMN4(CORLIN,CORSAM)  
C       STHE=STHE4(CORLIN,CORSAM)  
C       CALL 04U(0EOLAT,0EOLON,   STHE,STMN,STMCHO)

**DAN PACKAGE APPENDIX O  
COORDINATE TRANSFORMATIONS**

**04A  
002**

**RETURN  
END**

DAM PACKAGE APPENDIX O  
COORDINATE TRANSFORMATIONS

G4P  
001

SUBROUTINE G4P(    : GEOGRAPHIC COORDINATES FROM PRINT/PLOT COORDINATES  
O GEDLAT,GEDLON.   : LATITUDE, LONGITUDE (DEGREES)

I PPDLIN,PPDCOL)   : PRINT/PLOT LINE, COLUMN (DEVICE UNITS)  
-----

C  
C  
C  
C HISTORY  
C -----

C       E H SCHLOSSER       LEC       05/30/73       ORIGINAL CODE IN P20  
C       E H SCHLOSSER       LEC       10/31/79       RENAME G4P & REVISE ARG SEQUENCE

C  
C  
C METHOD  
C -----

C       TRANSFORM PRINT/PLOT COORDINATES TO CORRECTED COORDINATES.  
C       IF 'SPACING' THESE APPARENTLY CORRECTED COORDINATES ARE REALLY ADJUSTED  
C       SCANNER COORDINATES: TRANSFORM THEM TO TRUE CORRECTED COORDINATES.  
C       TRANSFORM CORRECTED COORDINATES TO SCENE TRANSVERSE MERCATOR COORDINATES.  
C       TRANSFORM SCENE TRANSVERSE MERCATOR COORDINATES TO GEOGRAPHIC COORDINATES

C  
C  
C MACHINE-DEPENDENT CODE  
C -----

C       NONE.

C  
C  
C EXTERNAL REFERENCES  
C -----

C       G4U                : GEOGRAPHIC COORDINATES FROM UTM (OR STM) COORDINATES

C  
C  
C GLOBAL DECLARATIONS  
C -----

C       INCLUDE KOMNER.LIST       : COMMON ERTS SCENE PARAMETERS  
C       INCLUDE KOMFIT.LIST       : COMMON ADJUSTMENT/REGISTRATION PARAMETERS  
C       INCLUDE TRFORM.LIST       : DEFINE COORDINATE TRANSFORMATION FUNCTIONS

C  
C  
C LOCAL DECLARATIONS  
C -----

C       REAL CORLIN,CORSAM       : CORRECTED LINE, SAMPLE  
C       REAL ADJLIN,ADJSAM       : ADJUSTED SCANNER LINE, SAMPLE  
C       DEFINE STMH=STMH+C(CORLIN,CORSAM)       : SCENE TRANSVERSE MERCATOR EASTING  
C       DEFINE STMN=STMN+C(CORLIN,CORSAM)       : SCENE TRANSVERSE MERCATOR NORTHING

C  
C  
C PROCEDURE  
C -----

C

DAN PACKAGE APPENDIX O  
COORDINATE TRANSFORMATIONS

04P  
002

```
      CORLIN=CORL4P(PPDLIN,PPDCOL)
      CORSAM=CORS4P(PPDLIN,PPDCOL)
C
C
C IF REPRESENTATIVE FRACTION IS DEFINED (DENOMINATOR NOT ZERO) DO 'SCALING'
C
C      IF(IIRFD.NE.0) GO TO 200      & 'SCALED' OUTPUT
C
C
C 'SPACED' OUTPUT -- PPD COORD NOT CORRECTED FOR ORBIT AND SCAN GEOMETRY
C
C      ADJLIN=CORLIN
C      ADJSAM=CORSAM
C      CORLIN=CORL4A(ADJLIN,ADJSAM)      & CORRECT COORD TO GET STM AND GEO
C      CORSAM=CORS4A(ADJLIN,ADJSAM)
C
C
C TRANSFORM CORRECTED TO SCENE TRANSVERSE MERCATOR TO GEOGRAPHIC COORDINATES
C
C 200 CALL 04U(GEDLAT,GEDLON, STME,STMN,STMCHD)      & STME, STMN DEFINED ABOVE
C
C      RETURN
C      END
```

04U/CLARKE 1068  
001

```

1 UTMH.UTMH.      8 UNIVERSAL TRANSVERSE MERCATOR EASTING. NORTHING (METRES)
1 UNDOQ)          8 PROJECTION CENTRAL MERIDIAN (DEGREES)

```

C	T R KELL	LEC	03/17/73	ORIGINAL CODE IN U20
C	E H SCHLOSSER	LEC	03/20/79	RENAME G4U & REVISE ARO SEQUENCE

**VALID ONLY FOR CLARKE 1866 SPHEROID.**

C THIS SOURCE CODE IS DESIGNED FOR A COMPILER THAT GENERATES SUBROUTINE  
C CODE WHICH PRESERVES THE VALUES OF LOCAL VARIABLES BETWEEN SUCCESSIVE  
C CALLS. WHEN A LARGE NUMBER OF POINTS ON A REGULAR GRID ARE TO BE  
C TRANSFORMED. GREATER EFFICIENCY WILL RESULT IF THEY ARE GROUPED BY EASTING  
C OR NORTHING.

**NONE.**

**NO.**

```

REAL FE      /500000./      3 FALSE EASTING
REAL UTMNL/-1000000000./    3 LAST NORTHING
REAL UTMEL/-1000000000./    3 LAST EASTING

```

```
IF(UTHE.EQ.UTHEL) GO TO 100
    UTHEL = UTHE
```

DAN PACKAGE APPENDIX O  
COORDINATE TRANSFORMATIONS

04U/CLARKE1866  
002

```

      Q=(UTME-FE)/1000000.
      Q2=Q*Q
      Q4=Q2*Q2
100 IF(UTMN.EQ.UTMNL)      GO TO 200
      UTMNL = UTMN
      OMSEC = .0324068466341*UTMN
      COM = COS(OMSEC*3.1415926536/(180.*3600.))
      COM2 = COM*COM
      COM4 = COM2*COM2
      SOM = SQRT(1.-COM2)
      PHISEC = OMSEC*
      &      (1047.546691+6.193011*COM2+.050699*COM4)*SOM*COM
200 CP=COS(PHISEC*3.1415926536/(180.*3600.))
      CP2 = CP*CP
      CP4 = CP2*CP2
      SP=SQRT(1.-CP2)
      SMCP2 = 1.+0.0068147849*CP2
      GEDLAT=PHISEC-2519.973189*Q2*(SMCP2)+.2*(SP/CP)*
1  (1.-.0020361958*Q2*(1.9591113+3./CP2+.081359*CP2+
2  .000279*CP4)+.00000165844*Q4*(SMCP2)+115.5
3  +45./CP4-.307/CP2+1.53*CP2))
      GEDLAT = GEDLAT / 3600.
      GEDLON=CMDEG*3600.-32242.2636*SQRT(SMCP2)/CP*Q*(1.-
4  .00407239159*SMCP2*Q2*(SMCP2-2.*2./CP2)+
5  .000004975312*SMCP2*SMCP2*Q4*(1.054+24./CP4
6  -20./CP2-.0138*CP2))
      GEDLON = GEDLON / 3600.
      RETURN
      END

```

FOIA  
001

```

C -----
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      05/24/73      ORIGINAL CODING (A2P)
C      D A BECK           LEC      10/23/79      RENAME/REVISE A4P
C
C METHOD
C -----
C
C      IF OUTPUT IS SCALED THEN
C          CORRECT FOR ORBIT AND SCAN GEOMETRY
C      TRANSFORM 'CORRECTED' TO PRINT/PLOT COORDINATES.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      NONE.
C
C EXCEPTIONS
C -----
C
C      1. IF PPD TRANSFORMATIONS ARE 'SPACED' AND CALSPA HAS NOT
C          BEEN CALLED THEN RESULTS ARE UNDEFINED.
C
C      2. IF PPD TRANSFORMATIONS ARE 'SCALED' AND CALSCA HAS NOT,
C          BEEN CALLED THEN RESULTS ARE UNDEFINED.
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOHNER.LIST      & COMMON ERTS SCENE PARAMETERS
C      INCLUDE KOMFIT.LIST      & COMMON ADJUSTMENT/REGISTRATION PAR
C      INCLUDE TRFORM.LIST      & DEFINE COORDINATE TRANSFORMATIONS
C
C LOCAL DECLARATIONS
C -----

```



DAN PACKAGE APPENDIX 0  
COORDINATE TRANSFORMATIONS

P4A  
002

```

C
C      REAL CORLIN      8 CORRECTED SCANNER LINE
C      REAL CORSAM      8 CORRECTED SCANNER SAMPLE
C
C
C PROCEDURE
C -----
C
C
C IF REPRESENTATIVE FRACTION IS DEFINED. DO SCALING ELSE DO SPACING
C
C      IF(IINFO.EQ.0) GO TO 200
C
C
C 'SCALED' OUTPUT -- CORRECT FOR ORBIT AND SCAN GEOMETRY
C
C      CORLIN=CORL4A(ADJLIN,ADJSAM)
C      CORSAM=CORS4A(ADJLIN,ADJSAM)
C      GO TO 300
C
C
C 'SPACED' OUTPUT -- DO NOT CORRECT FOR ORBIT AND SCAN GEOMETRY
C
C      200 CORLIN=ADJLIN
C          CORSAM=ADJSAM
C
C
C TRANSFORM 'CORRECTED' TO PRINT/PLOT COORDINATES
C
C      300 PPDLIN=PPDL4C(CORLIN,CORSAM)
C          PPOCOL=PPDC4C(CORLIN,CORSAM)
C
C
C RETURN TO CALLING ROUTINE
C
C      900 RETURN
C          END

```

PAGE  
001

```

C -----
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      09/29/73      ORIGINAL CODE IN 02P
C      E M SCHLOSSER      LEC      10/30/79      RENAME P40 & REVISE ARO SEQUENCE
C
C METHOD
C -----
C
C      TRANSFORM GEOGRAPHIC COORDINATES TO SCENE TRANSVERSE MERCATOR COORDINATES.
C      TRANSFORM SCENE TRANSVERSE MERCATOR COORDINATES TO CORRECTED COORDINATES.
C      IF 'SPACING' TRANSFORM CORRECTED COORDINATES TO ADJUSTED SCANNER
C      COORDINATES. AND ADJUSTED SCANNER COORDINATES TO PRINT/PLOT COORDINATES.
C      IF 'SCALING' TRANSFORM CORRECTED COORDINATES TO PRINT/PLOT COORDINATES.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      U40      3 UTM (OR STM) COORDINATES FROM GEOGRAPHIC COORDINATES
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMNER.LIST      3 COMMON ERTS SCENE PARAMETERS
C      INCLUDE KOMFIT.LIST      3 COMMON ADJUSTMENT/REGISTRATION PARAMETERS
C      INCLUDE TRFORM.LIST      3 DEFINE COORDINATE TRANSFORMATION FUNCTIONS
C
C LOCAL DECLARATIONS
C -----
C
C      REAL CORLIN.CORSAM      3 CORRECTED LINE. SAMPLE
C      REAL ADJLIN.ADJSAM      3 ADJUSTED SCANNER LINE. SAMPLE
C
C PROCEDURE
C -----
C
C      CALL U40(STME,STHN,  OEDLAT,OEDLON,STMCHD)
C      CORLIN=CORL4S(STME,STHN)

```

DAN PACKAGE APPENDIX 0  
COORDINATE TRANSFORMATIONS

P40  
002

```
      CORSAM=CORS4S(STHE,STHN)
C
C
C IF REPRESENTATIVE FRACTION IS DEFINED (DENOMINATOR NOT ZERO) DO 'SCALING'
C
C      IF(IRFD.NE.0) GO TO 200      & 'SCALED' OUTPUT
C
C
C 'SPACED' OUTPUT -- REMOVE CORRECTIONS FOR ORBIT AND SCAN GEOMETRY
C
C      ADJLIN=ADJL4C(CORLIN,CORSAM)
C      ADJSAM=ADJS4C(CORLIN,CORSAM)
C      CORLIN=ADJLIN
C      CORSAM=ADJSAM
C
C
C TRANSFORM 'CORRECTED' TO PRINT/PLOT COORDINATES
C
C      200 PPDLIN=PPDL4C(CORLIN,CORSAM)
C          PPDCOL=PPDC4C(CORLIN,CORSAM)
C
C      RETURN
C      END
```

DAN PACKAGE APPENDIX O  
COORDINATE TRANSFORMATIONS

U40/CLARKE1888  
001

SUBROUTINE U40:    : UTM COORDINATES FOR GEOGRAPHIC COORDINATES (SINGLE PREC  
: UTME,UTMN.       : UNIVERSAL TRANSVERSE MERCATOR EASTING, NORTHING (METRES)  
:                   :  
:    :    : GEOGRAPHIC (DEGREES) LATITUDE, LONGITUDE  
:    :    : PROJECTION CENTRAL MERIDIAN (DEGREES)  
:    :    : -----

C  
C  
C  
C HISTORY  
C -----

E M SCHLOSSER	LEC	02/13/73	ORIGINAL CODE IN 02U
E M SCHLOSSER	LEC	03/20/79	RENAME U40 & REVISE ARG SEQUENCE

C  
C  
C  
C METHOD  
C -----

THIS SUBROUTINE TRANSFORMS GEOGRAPHIC COORDINATES IN DEGREES TO UTM  
(UNIVERSAL TRANSVERSE MERCATOR) COORDINATES IN METERS, RELATIVE TO THE  
SPECIFIED CENTRAL MERIDIAN. ALL COMPUTATIONS ARE ON THE CLARKE 1888  
SPHEROID, WHICH IS STANDARD FOR ALL OF NORTH AMERICA. LATITUDE IS ASSUMED  
NORTH, AND MUST BE POSITIVE. LONGITUDE IS ASSUMED WEST AND MUST BE  
POSITIVE. THE USER MAY SPECIFY EITHER A STANDARD (183-6-ZONE) OR A  
NON-STANDARD CENTRAL MERIDIAN.

C  
C  
C  
C MACHINE-DEPENDENT CODE  
C -----

MAXIMUM ERROR IN THE TRANSFORMED COORDINATES IS 0.2 METER FOR POINTS  
WITH LATITUDES BETWEEN 0 AND 80 DEGREES AND LONGITUDES WITHIN 3 DEGREES  
OF THE CENTRAL MERIDIAN. (THIS ASSUMES ALL COMPUTATIONS IN SINGLE  
PRECISION ON A COMPUTER WITH A WORD SIZE OF AT LEAST 36 BITS.)

THIS SOURCE CODE IS DESIGNED FOR A COMPILER THAT GENERATES SUBROUTINE  
CODE WHICH PRESERVES THE VALUES OF LOCAL VARIABLES BETWEEN SUCCESSIVE  
CALLS. WHEN A LARGE NUMBER OF POINTS ON A REGULAR ORATICULE ARE TO BE  
TRANSFORMED, GREATER EFFICIENCY WILL RESULT IF THEY ARE GROUPED BY  
LATITUDE.

THIS SOURCE CODE IS DESIGNED TO TAKE ADVANTAGE OF THE OPTIMIZATION  
CAPABILITY OF THE UNIVAC FORTRAN V COMPILER, WHICH RECOGNIZES COMMON  
SUB-EXPRESSIONS AND GENERATES CODE TO PRE-COMPUTE AND STORE THEIR VALUES  
IN AVAILABLE REGISTERS. EXPLICIT SOURCE CODE OPTIMIZATION MAY BE  
APPROPRIATE FOR OTHER COMPILERS.

C  
C  
C  
C EXTERNAL REFERENCES  
C -----

NONE (OTHER THAN SYSTEM ROUTINES).

C  
C  
C  
C GLOBAL DECLARATIONS  
C -----

DAN PACKAGE APPENDIX 0  
COORDINATE TRANSFORMATIONS

U48/CLARKE1886  
002

```

C
C     NONE.
C
C
C LOCAL DECLARATIONS
C -----
C
      REAL FE      / .5E+6/      & CENTRAL MERIDIAN FALSE EASTING = 500000 METERS
      REAL OLDLAT / .0E+0/      & OLD LATITUDE (INITIALLY IMPOSSIBLE)
      DEFINE SECLAT=OEDLAT*.38E+4      & LATITUDE IN SECONDS
      DEFINE RADLAT=OEDLAT*.314159265E+1/.18E+3      & LATITUDE IN RADIANS
      DEFINE P=(CMDEG-OEDLON)*.38E+0      & DELTA LONGITUDE IN .0001 SECONDS
      REAL COS01      & COSINE**1
      REAL COS02      & COSINE**2
      REAL SINCOS      & SIN*COS
C
C
C PROCEDURE
C -----
C
C CALL COS & SQRT ONLY IF LATITUDE IS DIFFERENT FROM LAST POINT
C
      IF(OLDLAT.EQ.OEDLAT) GO TO 200
      OLDLAT=OEDLAT
      COS01=COS(RADLAT)
      COS02=COS(RADLAT)**2
      SINCOS=SQRT(1.-COS(RADLAT)**2)*COS(RADLAT)
      ECCFAC=1./SQRT(1.+ .68147849E-2*COS(RADLAT)**2)
200 CONTINUE
C
C
C COMPUTE EASTING
C
      UTMN=FE
      & *P      *.310151921E+6*COS01+ECCFAC
      & *1.
      & -P**2 *.381740509E-3
      & *1.-.2E+1*COS02-.681478E-2*COS02**2)
C
C
C COMPUTE NORTHING
C
      UTMN=.38876768E+2*(SECLAT
      & -(1.108288394E+4-.4483386E+1*COS02+.23558E-1*COS02**2)
      & *SINCOS)
      & *P**2 *.751829472E+4*SINCOS+ECCFAC
      & *1.
      & *P**2 *.1988703E-3
      & *(1.-.6E+1*COS02+.8133386E-1*COS02**2)
C
      RETURN
      END

```

**PREFACE TO APPENDIX P  
-----**

**THE SUBROUTINES IN THIS APPENDIX PROVIDE FORTRAN-CALLABLE INTERFACES  
TO THE UNIVAC EXEC-8 OPERATING SYSTEM.**

**THOSE SUBROUTINES BEGINNING WITH EA UTILIZE ASCII CHARACTER STRINGS.  
AND THOSE BEGINNING WITH ER UTILIZE FIELDATA CHARACTER STRINGS.**

**THE I- AND D- BANKS HAVE BEEN REVERSED IN MOST OF THESE SUBROUTINES  
IN ORDER TO PREVENT THE COLLECTOR FROM CONSTRUCTING INDIRECT LOAD  
TABLE ENTRIES FOR THEM. FOR THIS REASON, THEIR INCLUSION IN OVERLAY  
SEGMENTS SHOULD NEVER BE SPECIFIED EXPLICITLY, BUT ALWAYS LEFT TO  
BE DETERMINED IMPLICITLY BY THE COLLECTOR.**

**DAM PACKAGE APPENDIX P  
EXECUTIVE REQUESTS**

**APPENDIX-P  
001**

SPRT.SC DAM.PREFACE-P  
 SPRT.SC DAM.APPENDIX-P  
 SPRT.SC DAM.EAPRNT/DAM  
 SPRT.SC DAM.EAREAD/DAM  
 SPRT.SC DAM.ERCSE/DAM  
 SPRT.SC DAM.ERDATE/DAM  
 SPRT.SC DAM.ERERR/DAM  
 SPRT.SC DAM.EREXIT/DAM  
 SPRT.SC DAM.ERFACL/DAM  
 SPRT.SC DAM.ERFITH/DAM  
 SPRT.SC DAM.ERINFO/DAM  
 SPRT.SC DAM.ERIO/DAM  
 SPRT.SC DAM.ERLOW/DAM  
 SPRT.SC DAM.ERPCHA/DAM  
 SPRT.SC DAM.ERPCT/DAM  
 SPRT.SC DAM.ERPFS/DAM  
 SPRT.SC DAM.ERPRCA/DAM  
 SPRT.SC DAM.ERPRCN/DAM  
 SPRT.SC DAM.ERPRNT/DAM  
 SPRT.SC DAM.ERPRTA/DAM  
 SPRT.SC DAM.ERREAD/DAM  
 SPRT.SC DAM.ERREDA/DAM  
 SPRT.SC DAM.ERSUPS/DAM  
 SMSO.N .ERTRAN  
 SPRT.SC DAM.ERTSWP/DAM  
 SPRT.SC DAM.ERTWAT/DAM  
 SPRT.SC DAM.ERWAIT/DAM  
 SMSO.N .NERTRN  
 SPRT.SC DAM.SYPOET/DAM  
 SPRT.SC DAM.SYSADD/DAM  
 SPRT.SC DAM.SYSOET/DAM  
 SPRT.SC DAM.SYSPUT/DAM

. (0009) SET TABS 8 12 & 31  
 .  
 . PRINT IMAGE ON TTY OR LINE PRINTER (ASCII)  
 . READ IMAGE FROM TTY OR CARD READER (ASCII)  
 . SUBMIT EXEC COMMANDS (FIELDATA)  
 . RETURN SYSTEM DATE AND TIME (FIELDATA)  
 . ERRS TERMINATE PROGRAM  
 . TERMINATE PROGRAM IMMEDIATELY  
 . RETRIEVE FACILITIES ASSIGNMENT INFORMATION  
 . RETRIEVE FACILITIES ASSIGNMENT INFORMATION  
 . RETRIEVE SYSTEM/RUN/PROGRAM/FILE INFO  
 . INITIATE I/O  
 . INITIATE I/O & WAIT FOR COMPLETION  
 . WRITE IMAGE TO ALTERNATE PUNCH FILE (FIELDATA)  
 . RETRIEVE PART OF PROGRAM CONTROL TABLE (PCT)  
 . PROGRAM FILE SEARCH FOR INFO ON ELEMENT  
 . SET ALTERNATE PRINT FILE CONTROLS (FIELDATA)  
 . SET PRINT FILE CONTROLS (FIELDATA)  
 . PRINT IMAGE ON TTY OR LINE PRINTER (FIELDATA)  
 . WRITE IMAGE TO ALTERNATE PRINT FILE (FIELDATA)  
 . READ IMAGE FROM TTY OR CARD READER (FIELDATA)  
 . READ IMAGE FROM READ ALT FILE (FIELDATA)  
 . RETURN ACCUMULATED SUPS (200 USEC INCR) FROM PCT  
 . SUBMIT EXEC REQUESTS (UNIVAC SYSTEM ROUTINE)  
 . SWAP TAPE REELS OF MULTI-REEL FILE  
 . TIMED WAIT UP TO 30 SECONDS  
 . WAIT FOR COMPLETION OF I/O  
 . SUBMIT EXEC REQUESTS (UNIVAC SYSTEM ROUTINE)  
 . OUTPUT PROMPT RECORD & GET NEXT SYSIN RECORD  
 . ADD DISK SYMBOLIC FILE OR ELT TO SYSIN RUNSTREAM  
 . GET NEXT RECORD FROM SYSIN RUNSTREAM  
 . OUTPUT RECORD TO SYSOUT PRIMARY CRT/PRINTER FILE

ORIGINAL PAGE IS  
OF POOR QUALITY

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

EAPRNT/DAN  
001

```

.      SUBROUTINE EAPRNT( 8 PRINT STRING OF ASCII CHARACTERS
.      I NSPACE.         8 NUMBER OF LINES TO SPACE BEFORE PRINTING
.      I NMDS.           8 NUMBER OF 36-BIT WORDS IN STRING
.      I IMAGE)          8 STRING OF ASCII CHARACTERS, PACKED 4 PER WORD
.      -----
.
.      (E H SCHLOSSER)
.
.      THIS ASSEMBLER SUBROUTINE MAKES THE I/O FACILITIES OF ER APRINTS
.      DIRECTLY AVAILABLE TO A FORTRAN PROGRAM. IT IS INSENSITIVE TO WHETHER
.      THE PROCESSOR IS OPERATING IN THIRD-WORD OR QUARTER-WORD MODE.
.
.
S(00)      AXRS
EAPRNT*    LA      A1,0,X11      . SPACING
           LA      A2,1,X11      . NUMBER OF WORDS
           LSSL     A1,8          . SHIFT SPACING 8 BITS LEFT
           LA      A0,2,X11      . BUFFER ADDRESS
           SA,H1     A1,TEMP      . PACK
           SA,S3     A2,TEMP      . INTO
           LXI,H1    A0,TEMP      . A0
           ER        APRINTS     . PRINT
           J         4,X11       . RETURN
S(01)      BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
TEMP      RES      1
           END

```



DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

EAREAD/DAN  
001

```

.   SUBROUTINE EAREAD:  8 READ STRING OF ASCII CHARACTERS
.   8 S.               8 END-OF-FILE RETURN LABEL
.   8 IMAGE.           8 STRING OF ASCII CHARACTERS, PACKED 4 PER WORD
.   8 ISTAT)           8 STATUS WORD
.                       M1 = STATUS BITS (CONSULT UNIVAC PRM)
.                       M2 = NUMBER OF WDS READ (EXCL TRAILING BLANK WDS)
.   -----

```

. (E H SCHLOSSER)

. THIS ASSEMBLER SUBROUTINE MAKES THE I/O FACILITIES OF ER AREADS  
. DIRECTLY AVAILABLE TO A FORTRAN PROGRAM.

\$ (00)	AXRS	. BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES	
EAREAD	LA	A0.1.X11	. XM = BUFFER ADDRESS
	LXI	A0.(EOF)	. X1 = EOF BRANCH
	ER	AREADS	. READ CARD IMAGE
	SA	A0.*2.X11	. STORE STATUS
	J	4.X11	. NORMAL RETURN
EOF	SA	A0.*2.X11	. STORE STATUS
	J	0.X11	. EOF RETURN
	END		

DAM PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERCSF/DAM  
001

```

SUBROUTINE ERCSF( 8 SUBMIT FACILITY REQUEST CONTROL STATEMENT
0 IFSTAT.        8 FACILITY REQUEST STATUS WORD
1 JCSF)          8 STRING CONTAINING CONTROL STATEMENT (ENDS WITH ' . . '
-----

```

HISTORY

```

E H SCHLOSSER  LEC    02/28/74    ORIGINAL CODE
E H SCHLOSSER  LEC    06/18/74    TRACE IF MTRACE IS ON
E H SCHLOSSER  LEC    01/07/80    KOMXQT MACRO

```

METHOD

THIS ASSEMBLER SUBROUTINE MAKES THE FACILITIES OF ER CSFS DIRECTLY  
AVAILABLE TO A FORTRAN ROUTINE.

EXCEPTIONS

1. IF THE CONTENTS OF JCSF ARE NOT A VALID EXEC-8 CSFS FACILITY REQUEST  
CONTROL STATEMENT, THE CURRENT PROGRAM WILL ABORT.

GLOBAL DECLARATIONS

```

AXRS
KOMXQT
COMMON PROGRAM EXECUTION SWITCHES

```

LOCAL DECLARATIONS

```

S(00) . D-BANK
PF          FORM    12.6.18
CSFPKT      PF      1.0.0      . CSF PRINT PACKET
STAPKT      PF      1.2.0      . STATUS WORD PRINT PACKET
STATUS      RES      2         . STATUS WORD PRINT BUFFER
BIAS        '000000'
CF          FORM    10.6.6.6
CONST       CF      '888 . ' . 1.13.14
DOTSPA      EQUIP   CONST..H1  . ' . ' ENDS CONTROL STMT
ONE         EQUIP   CONST..S4  . 1
MAXWRD      EQUIP   CONST..S5  . WORD NO 13
MAXLEN      EQUIP   CONST..S6  . 14 WORDS

```

PROCEDURE

S(01) . 1-BANK

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERCSF/DAN  
002

ERCSF*	TNZ	MTRACE	. CHECK TRACE SWITCH
	J	SUBMIT	. SWITCH IS OFF
	LA	A0.1.X11	. ADDR OF CONTROL STMT
	SA.M2	A0.CSFPKT	. INSERT IN PRINT PKT
	LXI	A0.ONE	. INCREMENT = 1
	LA	A3.MAXHRD	. MAXIMUM WORD NUMBER
LOAD2	DL	A1.0.*A0	. LOAD 2 WDS & INCREMENT A0
ALIGN	DSL	A1.18	. RIGHT ALIGN 3 LEADING CHARS
	TNE	A1.00TSPA	. CHECK FOR STMT END
	J	PRINT	. END FOUND
	LDSL	A1.24	. DELETE 1ST CHAR & LEFT ALIGN
	JNZ	A1.ALIGN	. TRY NEXT CHARS. IF ANY
	JOD	A3.LOAD2	. TRY NEXT WORD. IF ANY
PRINT	ANA	A3.MAXLEN	. ACTUAL STMT LENGTH
	SHA.S3	A3.CSFPKT	. INSERT IN PRINT PKT
	LA	A0.CSFPKT	. LOAD PRINT PACKET
	ER	PRINTS	. PRINT CONTROL STMT
SUBMIT	LA	A0.1.X11	. ADDR OF CONTROL STMT
	ER	CSFS	. SUBMIT CONTROL STATEMENT
	SA	A0.*0.X11	. PUT STATUS WORD IN 1ST ARG
	TNZ	MTRACE	. CHECK TRACE SWITCH
	J	3.X11	. RETURN
	LA	A1.BIAS	. '0' IS NEGATIVE!
	LA	A2.BIAS	.
	LA	A3.*0.X11	. STATUS WD
UNPACK	LDSL	A1.6	. SHIFT & PAD
	SSL	A2.3	. TOO FAR!
	LDSL	A2.3	. EXTRACT 3-BIT OCTAL DIGIT
	JN	A1.UNPACK	. REPEAT TIL LEADING CHAR NOT '0'
	AA	A1.BIAS	. CONVERT 6-BIT OCTAL
	AA	A2.BIAS	. TO FIELDATA
	DS	A1.STATUS	. PUT IN PRINT BUFFER
	LA	A0.STAPKT	. LOAD PRINT PKT
	LXM	A0.(STATUS)	. XM = ADDR OF STATUS WD
	ER	PRINTS	. PRINT STATUS WD
	J	3.X11	. RETURN
	END		

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERDATE/DAN  
001

```

.   SUBROUTINE ERDATE( 8 RETURN SYSTEM DATE & TIME (FIELDATA)
.   0 INDY.           8 MONTH DAY YEAR (2 CHARS EACH)
.   0 IHMS)           8 HOUR MINUTE SECOND (2 CHARS EACH)
.   -----
.
.   (M. L. BROWN)
.
.   THIS IS AN ASSEMBLER ROUTINE TO RETURN WITH THE DATE
.   AND THE TIME
.
$100)      AXRS      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
ERDATE*    LMJ      AO.ATRACE      . TRACE
.           . 'ERDATE'      . NAME OF THIS SUBROUTINE
.           ER        DATES        . REQUEST DATE & TIME
.           SA        AO.'0.X11    . MMDDYY
.           SA        A1.'1.X11    . HMMSS
.           J         3.X11        . RETURN
.           END

```

DAM PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERERR/DAM  
001

. SUBROUTINE ERERR 3 ERROR TERMINATE PROGRAM EXECUTION  
.  
.  
.

9(00)  
ERERR\*

AXRS  
ER  
END

. BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES  
ERRS

DAM PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

EREXIT/DAM  
001

```
. SUBROUTINE EREXIT 8 NORMALLY TERMINATE PROGRAM EXECUTION
. -----
.
.
$100) AXRS . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
EREXIT+ ER EXITS
END
```

DAM PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERFACL/DAM  
001

```

.      SUBROUTINE ERFACL( 8 RETRIEVE FACILITIES ASSIGNMENT INFORMATION
.      U LUPKT)          8 PACKET (MUST BE 8 WORDS)
.      -----
.
.      (E H SCHLOSSER)
.
.      THIS ASSEMBLER SUBROUTINE USES EXECUTIVE REQUEST FACILS TO RETURN THE
.      QUALIFIER NAME AND FACILITIES ASSIGNMENT INFORMATION FOR A CURRENTLY ASSIGNED
.      FILE. THE CALLING PROGRAM MUST PROVIDE A PROPERLY INITIALIZED PACKET
.      AS FOLLOWS:
.
.      DIMENSION PACKET-NAME(8)
.      .
.      PACKET-NAME(1)='FIRST-8-CHARS-OF-INTERNAL-FILE-NAME'
.      PACKET-NAME(2)='NEXT-8-CHARS-OF-INTERNAL-FILE-NAME-OR-BLANKS'
.
.      THE FORTRAN CALLING SEQUENCE FOR THIS SUBROUTINE THEN IS AS FOLLOWS:
.
.      CALL ERFACL(PACKET-NAME)
.
.
S(00)      AXRS      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
ERFACL*    LMJ       AD.ATRACE      . TRACE
.          .          'ERFACL'      . NAME OF THIS SUBROUTINE
.          L.U       AO.*0.X11     . PACKET ADDRESS
.          ER        FACILS        . FILL PACKET
.          J         Z.X11         . RETURN
.          END

```

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERFITH/DAN  
001

```

. SUBROUTINE ERFITH( 8 RETRIEVE FACILITIES ASSIGNMENT INFORMATION
. U LUPKT)          8 PACKET (MUST BE 13 WORDS)
. -----
.
. (E H SCHLOSSER)
.
. THIS ASSEMBLER SUBROUTINE USES EXECUTIVE REQUEST FITENS TO RETURN THE
. QUALIFIER NAME AND FACILITIES ASSIGNMENT INFORMATION FOR A CURRENTLY ASSIGNED
. FILE. THE CALLING PROGRAM MUST PROVIDE A PROPERLY INITIALIZED PACKET
. AS FOLLOWS:
.
.   DIMENSION PACKET-NAME(13)
.   .
.   PACKET-NAME(1)='FIRST-8-CHARS-OF-INTERNAL-FILE-NAME'
.   PACKET-NAME(2)='NEXT-8-CHARS-OF-INTERNAL-FILE-NAME-OR-BLANKS'
.
. THE FORTRAN CALLING SEQUENCE FOR THIS SUBROUTINE THEN IS AS FOLLOWS:
.
.   CALL ERFITH(PACKET-NAME)
.
.
. S(00)
. ERFITH*
.
. AXRS
. LMJ
.
. LA
. LXI
. ER
. J
. END
.
. BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
. A0.ATRACE . TRACE
. 'ERFITH' . NAME OF THIS SUBROUTINE
. A0.0.X11 . XM = PACKET ADDRESS
. A0.(037777) . XI = MAX PACKET LENGTH
. FITENS . FILL PACKET
. 2.X11 . RETURN

```



```

SUBROUTINE ERINFO(      8 RETRIEVE SYSTEM.RUN.PROGRAM.& FILE PARAMETERS
0 1STFIL.              8 STATUS OF FILE
                        0 = NORMAL COMPLETION
                        1 = ERROR IN ERINFO
                        2 = ILLEGAL FUNCTION
                        3 = FUNCTION NOT AVAILABLE
                        4,5,6,7 = ERROR IN ERINFO
                        8 = FILE NOT ASSIGNED TO RUN
0 1STFUN.              8 STATUS OF FUNCTION
                        (SEE ABOVE EXCEPT FOR)
                        6 = SPECIFIED PKT TOO SMALL - INVALID COUNT
                        7 = SPECIFIED PKT TOO LARGE - VALID COUNT
                        8 = INCORRECT INPUT FUNCTION/DATA.
                        9 = ERROR IN ERINFO
0 KNTFUN.              8 NO. OF WORDS REQUIRED TO TRANSFER ALL DATA TO PKT.
0 1PKT.                8 PACKET FOR FUNCTION
1 LENPKT.              8 LENGTH IN WORDS OF PACKET
                        =
1 NAMFIL.              8 NAME OF FILE - PADDED WITH BLANKS TO 12 CHARS
1 NUMFUN)              8 FUNCTION CODE NUMBER
-----

```

#### HISTORY

-----

M A TOMPKINS      LEMSCO      09/15/80      ORIGINAL CODE

#### METHOD

-----

RETRIEVE SYSTEM/RUN/PROGRAM/FILE PARAMETERS VIA ER INFOS AS FOLLOWS:  
GENERATE AN INTERNAL 2-WORD FUNCTION STRING FOR THE USER-SPECIFIED  
FUNCTION CODE NUMBER AND PACKET. IF FILENAME IS SPECIFIED (NOT NULL)  
PREFIX INTERNAL FUNCTION STRING WITH AN INFILES FUNCTION STRING.  
PERFORM ER. RETRIEVE STATUS AND COUNT FROM INTERNAL FUNCTION STRING.  
CALL ENTRYPT (LUPKT,IFUNC)

#### MACHINE-DEPENDENT CODE

-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 8-BIT  
FIELDATA CHARACTERS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.  
DIFFERENT COMPILERS (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

#### EXTERNAL REFERENCES

-----

INFIOB    4 EXECUTIVE REQUEST ROUTINE TO RETRIEVE SYSTEM.  
          RUN.PROGRAM.& FILE PARAMETERS

. EXCEPTIONS

1. IF THE FILENAME IS NULL THEN THE FILE STATUS (1STFIL) WILL BE  
UNDEFINED.

. GLOBAL DECLARATIONS

AXRS

. STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS

\$1001 . D-BANK

INFLF	FORM	12.6.10	. FUNCTION.STATUS.COUNT
FUNSTR	INFLF	3.0.0	. FUNCTION = INFILS
		2.0	. PKT LENGTH = 2 WORDS
	RES	2	. FOR NEXT FUNCTION

. PROCEDURE

\$1011 . 1-BANK

ERINFO	LMJ	A0.ATRACE	. TRACE
		'ERINFO'	. NAME OF THIS SUBROUTINE
	LA	A0.(FUNSTR)	. XM = FUNCTION STRING ADDRESS INFILS
	LXI.U	A0.4	. XI = MAX STRING LENGTH IN WORDS
	LA	A4.*5.X11	. CHECK FILENAME
	JNZ	*A4.MAX	. TEST - JUMP IF VALID FILENAME
	LA	A0.(FUNSTR+2)	. XM = FUNCTION STRING NO INFILS
	LXI.U	A0.2	. XI = MIN STRING LENGTH IN WORDS
MAX	LA	A1.5.X11	. FILENAME ADDRESS
	SA.M2	A1.FUNSTR+1	. STORE FILENAME ADDRESS WORD 2 OF FUNSTR
	LXI	A2.*6.X11	. FUNCTION CODE NUMBER
	LSSL	A2.6	. SHIFT NUNFUN 6 BITS LEFT
	SA	A2.FUNSTR+2	. STORE IN S1 OF WORD 3 OF FUNSTR
	LA	A3.3.X11	. PKT ADDRESS
	LXI	A3.*4.X11	. PKT LENGTH
	SA	A3.FUNSTR+3	. STORE IN WORD 4 OF FUNSTR
	ER	INFOS	. GET INFORMATION
	LA.S3	A0.FUNSTR	.
	SA	A0.*0.X11	. STORE FILE STATUS
	LA.S3	A0.FUNSTR+2	.
	SA	A0.*1.X11	. STORE FUNCTION STATUS

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERINFO/DAN  
003

LA.M2	AG.FUNSTR*2	.
SA	AG.*2.X11	. STORE WORD COUNT OF FUNCTION
J	0.X11	. RETURN
END		

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERIO/DAN  
891

```

SUBROUTINE ERIO( 8 INITIATE I/O AND RETURN BEFORE COMPLETION
U LUPKT)          8 PACKET (MUST BE 8 WORDS)
-----
(IE M SCHLOSSER)

THIS ASSEMBLER SUBROUTINE MAKES THE I/O FACILITIES OF ER IOB DIRECTLY AVAIL-
ABLE TO A FORTRAN PROGRAM. THE CALLING PROGRAM MUST PROVIDE A BUFFER AND
A PROPERLY INITIALIZED PACKET AS FOLLOWS:

    INCLUDE KOMIO.LIST
    DIMENSION BUFFER-NAME(BUFFER-SIZE)
    DIMENSION PACKET-NAME(8)      8 ALWAYS 8 WORDS!!
    .
    .
    PACKET-NAME(1)='FIRST-8-CHARS-OF-INTERNAL-FILE-NAME'
    PACKET-NAME(2)='NEXT-8-CHARS-OF-INTERNAL-FILE-NAME-OR-BLANKS'
    .
    .
    IOSIZE(PACKET-NAME)=BUFFER-SIZE
    IOADDR(PACKET-NAME)=LOC(BUFFER-NAME)
    IOWAIT(PACKET-NAME)=MILLISECONDS-TO-WAIT      8 TAPE ONLY (MAX 83)
    .
    .
    * SYNCHRONIZATION BUGS IN THE UNIVAC-SUPPLIED TAPE I/O HANDLER WITHIN THE
    * EXEC CAUSE PREMATURE SETTING OF THE I/O STATUS TO COMPLETE, RESULTING
    * IN SPORADIC TAPE MALFUNCTIONS WHEN THE SYSTEM IS LIGHTLY LOADED. THE
    * IOWAIT CONSTRUCT ALLOWS THE USER TO SPECIFY (FOR TAPE I/O ONLY) A TIMED
    * WAIT OF 0 THRU 83 MILLISECONDS TO BE TAKEN BY SUBROUTINE ERWAIT WHEN THE
    * STATUS INDICATES I/O IS ALREADY COMPLETE ON ENTRY TO ERWAIT. IOWAIT
    * SHOULD NORMALLY BE SET TO 10 MILLISECONDS WHEN NO OTHER EXEC REQUEST
    * INTERVENES BETWEEN THE CALL TO ERIO AND THE CALL TO ERWAIT.
    .
    .
    THE CALLING SEQUENCE FOR THIS SUBROUTINE THEN IS AS FOLLOWS:

    IOSECT(PACKET-NAME)=FILE-RELATIVE-SECTOR-ADDRESS      8 FASTRAND ONLY
    IOWORD(PACKET-NAME)=FILE-RELATIVE-WORD-ADDRESS        8 DRUM ONLY
    IOFUNC(PACKET-NAME)='FUNCTION-LITERAL'                8 SEE BELOW
    CALL ERIO(PACKET-NAME)
    .
    .
    CALL ERWAIT(PACKET-NAME)      8 WAIT UNTIL I/O IS COMPLETE
    .
    .
    THE FOLLOWING DEFINE FUNCTIONS IN THE KOMIO PROC RETURN I/O COMPLETION
    INFORMATION:

    IOSTAT(PACKET-NAME)      I/O (TYPE 1) STATUS CODE.
    IOAFCT(PACKET-NAME)      ABNORMAL FRAME COUNT (TAPE ONLY)
    IONWOS(PACKET-NAME)      NUMBER OF WORDS READ/Written
    .
    .
    LITERALS FOR COMMON I/O FUNCTIONS AND EQUIPMENT TYPES ARE AS FOLLOWS:

    LITERAL    OCTAL    MNEMONIC    TYPES    MEANING
    .
    'BC'       10       MS           T F 0      WRITE
    .
    'BD'       11       MEFS         T           WRITE EOF (TAPE)
    .
    'BK'       20       RS           T F 0      READ
    .
    'BL'       21       RBS           T           READ BACKWARD

```

DAM PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ER10/DAM  
002

.	'8H'	22	RRS	F	READ & RELEASE
.	'8N'	23	RELS	F	RELEASE
.	'8O'	24	BRDS	D	BLOCK READ DRUM
.	'8P'	25	ROLS	F D	READ & LOCK
.	'8Q'	26	UNLS	F D	UNLOCK
.	'8)	40	REWS	T	REWIND
.	'8-	41	REWS	T	REWIND WITH INTERLOCK
.	'8+	42	SMS	T	SET MODE
.			MFS	T	MOVE FORWARD (DO NOT USE!!!)
.			MS	T	BACKSPACE (DO NOT USE!!!)

S(01)  
ER10\*

AXRS  
L.U  
ER  
J  
END

A0.\*0.X11  
10S  
2.X11

. PACKET ADDRESS  
. REQUEST I/O INITIATION  
. RETURN

```

SUBROUTINE ERLOW( 8 INITIATE I/O AND WAIT FOR COMPLETION
U LUPKT)          8 PACKET (MUST BE 8 WORDS)
-----
(E M SCHLOSSER)

THIS ASSEMBLER SUBROUTINE MAKES THE I/O FACILITIES OF ER LOWS DIRECTLY AVAIL-
ABLE TO A FORTRAN PROGRAM. THE CALLING PROGRAM MUST PROVIDE A BUFFER AND
A PROPERLY INITIALIZED PACKET AS FOLLOWS:

    INCLUDE KOMIO.LIST
    DIMENSION BUFFER-NAME(BUFFER-SIZE)
    DIMENSION PACKET-NAME(8)      8 ALWAYS 8 WORDS!!
    .
    PACKET-NAME(1)='FIRST-8-CHARS-OF-INTERNAL-FILE-NAME'
    PACKET-NAME(2)='NEXT-8-CHARS-OF-INTERNAL-FILE-NAME-OR-BLANKS'
    .
    IOSIZE(PACKET-NAME)=BUFFER-SIZE
    IOADDR(PACKET-NAME)=LOC(BUFFER-NAME)
    IOWAIT(PACKET-NAME)=MILLISECONDS-TO-WAIT      8 TAPE ONLY (MAX 63)

    * SYNCHRONIZATION BUGS IN THE UNIVAC-SUPPLIED TAPE I/O HANDLER WITHIN THE
    * EXEC CAUSE PREMATURE SETTING OF THE I/O STATUS TO COMPLETE, RESULTING
    * IN SPORADIC TAPE MALFUNCTIONS WHEN THE SYSTEM IS LIGHTLY LOADED. THE
    * IOWAIT CONSTRUCT ALLOWS THE USER TO SPECIFY (FOR TAPE I/O ONLY) A TIMED
    * WAIT OF 0 THRU 63 MILLISECONDS TO BE TAKEN BY SUBROUTINE ERLOW AFTER THE
    * STATUS INDICATES I/O IS COMPLETE. IOWAIT SHOULD NORMALLY BE SET TO 10
    * MILLISECONDS WHEN NO OTHER EXEC REQUEST INTERVENES BETWEEN SUCCESSIVE
    * CALLS TO ERLOW FOR THE SAME TAPE FILE.

    .
    THE CALLING SEQUENCE FOR THIS SUBROUTINE THEN IS AS FOLLOWS:

    IOSECT(PACKET-NAME)=FILE-RELATIVE-SECTOR-ADDRESS      8 FASTRAND ONLY
    IOWORD(PACKET-NAME)=FILE-RELATIVE-WORD-ADDRESS        8 DRUM ONLY
    IOFUNC(PACKET-NAME)='FUNCTION-LITERAL'                8 SEE SUBROUTINE ERIO
    CALL ERLOW(PACKET-NAME)

    .
    THE FOLLOWING DEFINE FUNCTIONS IN THE KOMIO PROC RETURN I/O COMPLETION
    INFORMATION:

    IOSTAT(PACKET-NAME)      I/O (TYPE 1) STATUS CODE.
    IOAFCT(PACKET-NAME)      ABNORMAL FRAME COUNT (TAPE ONLY)
    IONWDS(PACKET-NAME)      NUMBER OF WORDS READ/Written

    .
    $ (01)
    ERLOW*      AXRS
                L.U      A0.0.X11      . PACKET ADDRESS
                LA.S1    A1.5.A0      . MILLISECONDS TO WAIT
                ER        IOWS      . REQUEST I/O INITIATION & COMPLETION
                TZ        A1      . SKIP NI IF MILLISECONDS = 0
                ER        TWAITS    . TIMED WAIT FOR A1 MILLISECONDS
                J          2.X11      . RETURN
                END

```

DAM PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERPCHA/DAM  
001

```

SUBROUTINE ERPCHA( 8 WRITE FIELDATA CHARACTER STRING TO ALT PUNCH FILE
1 NAMFIL.          8 NAME OF ALT PUNCH FILE (12 CHARACTERS, BLANK FILLED)
1 NMDS.            8 NUMBER OF 36-BIT WORDS IN STRING
1 IMAGE)           8 STRING OF FIELDATA CHARACTERS, PACKED 8 PER WORD
-----

```

(E H SCHLOSSER)

THIS ASSEMBLY SUBROUTINE MAKES THE I/O FACILITIES OF ER PNCAS  
DIRECTLY AVAILABLE TO A FORTRAN PROGRAM.

```

$00)      AXRS      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
ERPCHA*   DL        A2.0.X11      . GET ALT PCH FILE NAME
          LXI       A1.01.X11     . GET NUMBER OF WORDS
          LA        A0.(PKT)      . PUT ADDRESS OF PACKET IN A0
          LXM       A1.2.X11     . GET BUFFER ADDRESS
          DS        A2.PKT+1      . PUT FILE NAME IN PKT+1, PKT+2
          SA        A1.PKT        . PUT WD-COUNT/BUFFER-ADDR IN PKT+0
          ER        PNCAS         . WRITE BUFFER TO ALT PUNCH FILE
          J         4.X11         . RETURN

```

```

$01)
PKT      RES      3
          END

```

DAM PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERPCT/DAM  
001

. SUBROUTINE ERPCT: 3 RETRIEVE PART OF PROGRAM CONTROL TABLE  
. 1 NHDS. 3 NUMBER OF WORDS TO BE RETRIEVED FROM BEGINNING OF PCT  
. 0 JPCT) 3 ARRAY NHDS LONG USED TO RETURN REQUESTED PART OF PCT  
-----

. (E H SCHLOSSER)

. THIS ASSEMBLER SUBROUTINE RETURNS THE FIRST N WORDS OF THE PCT (PROGRAM  
. CONTROL TABLE).

. THE PCT IS SUBJECT TO CHANGE. HOWEVER THE CURRENT (LEVEL 31) ADDRESSES OF  
. INFORMATION IN THE PCT USED BY THE DAM PACKAGE ARE AS FOLLOWS:

. WORD PART CONTENTS  
-----

. 1 ORIGINAL RUNID

. 2 GENERATED RUNID

. 4 ESTIMATED SUPS FOR RUN (200 USEC INCREMENTS)

. 12 ACCUMULATED SUPS FOR RUN (200 USEC INCREMENTS)

. 15 RUN START DATE/TIME:

. S1 MONTH  
. S2 DATE  
. S3 (YEAR-1984)  
. H2 SECONDS SINCE MIDNIGHT

. 16 3XQT OPTIONS:

. A B C D E F G H I J K L M N O P Q R S . Z  
. BIT: 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 .. 00

. 17 CONDITION WORD

. 18-19 QUALIFIER

. 20-21 ACCOUNT NUMBER

. 25 S2 PROGRAM TYPE:  
. 2-REAL TIME  
. 3-LOW EXEC  
. 4-DEMAND  
. 5-DEADLINE BATCH  
. 6-BATCH

S(00)  
ERPCT: AXRS . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES  
LMJ A0.ATRACE . TRACE  
LA A0.1.X11 . NAME OF THIS SUBROUTINE  
LXI A0.0.X11 . XM = ADDRESS OF ARRAY  
ER PCTS . XI = NUMBER OF WORDS  
J 3.X11 . REQUEST IT  
END . RETURN



ERP/S/DAN  
001

**P-20**

DAM PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERPRCA/DAM  
001

```

.      SUBROUTINE ERPRCA( 3 SET ALTERNATE PRINT FILE CONTROLS (FIELDATA)
.      I NHWORS.          3 NUMBER OF WORDS IN PRINT CONTROL STRING
.      I JPRCON)          3 PRINT CONTROL STRING
.      -----
.
.      (E H SCHLOSSER)
.
.      THIS ASSEMBLER SUBROUTINE MAKES THE PRINT CONTROL FACILITIES OF ER PRTCAS
.      DIRECTLY AVAILABLE TO A FORTRAN PROGRAM.
.
.
$100)      AXRS      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
ERPRCA*    LHJ      A0.ATRACE      . TRACE
.          'ERPRCA'      . NAME OF THIS SUBROUTINE
.          A0.1.X11      . XM = ADDR OF PRT CONTROL STRING
.          A0.0.X11      . XI = NO OF WDS IN STRING
.          PRTCAS      . ALT PRINT CONTROL
.          3.X11      . RETURN
.          J
.          END

```

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERPRCN/DAN  
001

```

      AXRS .
. REPLACES EXEC-11 *SETHRO.PAPER.PAP30***8H00.LINE ADDED
. ---- UP-4144 REV 2.EXEC 8 PRM. 8 5.4.1 (ER PRTCHS) 3/19/73 VLR
. ---- PAGES 5-11,12 OR **UP-7024 PAGE2-33** ---- 8FOR CALL IS
. CALL ERPRCN (NR-OF-WORDS-IN-STRING,'PRINT-CONTROL-STRING-TEXT')
ERPRCN*  L,U      XII.0,X11
        L        A0.1,X11      . ADDRESS OF TEXT TO PRTCHS
        LXI      A0.0,X11      . NR OF WORDS OF TEXT
        ER       PRTCHS .      . DO IT
        J        3.X11        . RETURN

END .

```

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERPRINT/DAN  
001

```

.      SUBROUTINE ERPRINT( 8 PRINT STRING OF FIELDATA CHARACTERS
.      I NSPACE,          8 NUMBER OF LINES TO SPACE BEFORE PRINTING
.      I NMOS,            8 NUMBER OF 36-BIT WORDS IN STRING
.      I IMAGE)           8 STRING OF FIELDATA CHARACTERS, PACKED 8 PER WORD
.      -----
.
.      (E H SCHLOSSER)
.
.      THIS ASSEMBLER SUBROUTINE MAKES THE I/O FACILITIES OF ER PRINTS
.      DIRECTLY AVAILABLE TO A FORTRAN PROGRAM. IT IS INSENSITIVE TO WHETHER
.      THE PROCESSOR IS OPERATING IN THIRD-WORD OR QUARTER-WORD MODE.
.
.
S(00)      AXRS      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
ERPRINT+   LA        A1,*0.X11      . SPACING
           LA        A2,*1.X11      . NUMBER OF WORDS
           LSSL      A1,8           . SHIFT SPACING 8 BITS LEFT
           LA        A0,2.X11      . BUFFER ADDRESS
           SA,M1      A1,TEMP       . PACK
           SA,S3      A2,TEMP       . INTO
           LXI,M1     A0,TEMP       . XI
           ER         PRINTS        . PRINT
           J          4.X11        . RETURN

S(01)
TEMP      RES      1
           END

```

DAM PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERPRTA/DAM  
001

```

. SUBROUTINE ERPRTA( 8 WRITE STRING OF FIELDATA CHARACTERS TO ALT PRT FILE
. 1 NAMFIL.          8 NAME OF ALT PRINT FILE (12 CHARACTERS, BLANK FILLED)
. 1 NSPACE.          8 NUMBER OF LINES TO ADVANCE BEFORE PRINTING
. 1 NMDS.            8 NUMBER OF 36-BIT WORDS IN STRING
. 1 IMAGE)           8 STRING OF FIELDATA CHARACTERS, PACKED 8 PER WORD
. -----

```

(E H SCHLOSSER)

THIS ASSEMBLER SUBROUTINE MAKES THE I/O FACILITIES OF ER PRTAS  
DIRECTLY AVAILABLE TO A FORTRAN PROGRAM. IT IS INSENSITIVE TO WHETHER  
THE PROCESSOR IS OPERATING IN THIRD-WORD OR QUARTER-WORD MODE.

```

$ (00)      AXRS      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
ERPRTA*     DL        A2,*0,X11      . GET ALT PRT FILE NAME
            LA        A0,*1,X11      . GET SPACING
            OS        A2,PKT+1       . PUT FILE NAME IN PKT+1, PKT+2
            LSSL      A0,8           . SHIFT SPACING 8 BITS LEFT
            LA        A1,*2,X11      . GET NUMBER OF WORDS
            LA        A2,3,X11       . GET BUFFER ADDRESS
            SA,M1      A0,PKT         . PACK
            SA,S3      A1,PKT         . INTO
            SA,M2      A2,PKT         . PKT+0
            LA        A0,(PKT)        . ADDRESS OF PACKET
            ER        PRTAS          . PRINT
            J         S,X11          . RETURN

$ (01)
PKT         RES       3
            END

```

DAM PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERREAD/DAM  
001

```

.      SUBROUTINE ERREAD:  & READ STRING OF FIELDATA CHARACTERS
.      & S.              & END-OF-FILE RETURN LABEL
.      & IMAGE.          & STRING OF FIELDATA CHARACTERS. PACKED & PER WORD
.      & ISTAT:          & STATUS WORD:
.                          M1 = STATUS BITS (CONSULT UNIVAC PRM)
.                          M2 = NUMBER OF WDS READ (EXCL TRAILING BLANK WDS)
.      -----
.
. (E H SCHLOSSER)
.
. THIS ASSEMBLER SUBROUTINE MAKES THE I/O FACILITIES OF ER READS
. DIRECTLY AVAILABLE TO A FORTRAN PROGRAM.
.
$1001      AXRS      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
ERREAD*    LA        A0.1.X11      . XM = BUFFER ADDRESS
          LXI        A0.(EOF)      . XI = EOF BRANCH
          ER         READS         . READ CARD IMAGE
          SA         A0.*2.X11     . STORE STATUS
          J          4.X11         . NORMAL RETURN
EOF        SA         A0.*2.X11     . STORE STATUS
          J          0.X11         . EOF RETURN
          END

```

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERREDA/DAN  
001

```

.      SUBROUTINE ERREDA( 8 READ STRING OF FIELDATA CHARACTERS FROM ALT FILE
.      1 NAMFIL.         8 NAME OF READ ALT FILE (12 CHARS. BLANK FILLED)
.      6 S.              8 END-OF-FILE RETURN LABEL
.      0 IMAGE.          8 STRING OF FIELDATA CHARACTERS. PACKED 6 PER WORD
.      0 ISTAT)          8 STATUS WORD:
.                          H1 = STATUS BITS (CONSULT UNIVAC PRM)
.                          H2 = NUMBER OF WDS READ (EXCL TRAILING BLANK WDS)
.      -----
.      (E H SCHLOSSER)
.
.      THIS ASSEMBLER SUBROUTINE MAKES THE I/O FACILITIES OF ER READAS
.      DIRECTLY AVAILABLE TO A FORTRAN PROGRAM.
.
S(00)      AXRS      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
ERREDA*    OL        A2.*0.X11      . GET ALT READ FILE NAME
.          LA        A1.2.X11      . XM = BUFFER ADDRESS
.          LXI       A1.(EOF)      . XI = EOF BRANCH
.          DS        A2.PKT+1      . PUT FILE NAME IN PKT+1. PKT+2
.          SA        A1.PKT        . PUT EOFADDR/BUFADDR IN PKT+0
.          LA        A0.(PKT)      . ADDRESS OF PKT
.          ER        READAS        . READ ALTERNATE CARD IMAGE
.          SA        A0.*3.X11     . STORE STATUS
.          J         S.X11         . NORMAL RETURN
EOF        SA        A0.*3.X11     . STORE STATUS
.          J         I.X11         . EOF RETURN

S(01)
PKT        RES      3
.          END

```

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERSUPS/DAN  
001

```

.   SUBROUTINE ERSUPS:  0 RETURN ACCUMULATED SUP TIME FOR CURRENT RUN
.   0 NSUPS)           0 ACCUMULATED SUP TIME (200 USEC INCREMENTS)
.   -----
.
.   (E H SCHLOSSER)
.
.
S(00)  AXRS  . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES
ERSUPS L,U   A0,*0.X11  . BUFFER ADDRESS
        LA   A1,(1,11)  . 1 WORD, STARTING AT PCT+11 (-PCT(12))
        ER   PCTS
        J    2.X11
        END

```



DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERTSHP/DAN  
001

```

SUBROUTINE ERTSHP (      0 CLOSE CURRENT REEL FOR TAPE FILE AND
                        0 REQUEST LOADING OF ANOTHER REEL
I LUPKT.      0 PACKET (MUST BE 3 WORDS)
I IFUNC)      0 SWAP OPTION INDICATOR
              0      0 = SWAP TO NEXT REEL
              0      1 = SWAP TO NEXT REEL & STORE REEL # IN LUPKT(3)
              0      2 = MOUNT REEL SPECIFIED IN LUPKT(3)
-----
.
.
.
HISTORY
-----
.
.
.
C A HELMKE      LEC      09/03/79      REQUIREMENTS
C A HELMKE      LEC      09/03/79      ALGORITHM DESIGN
C A HELMKE      LEC      09/03/79      ALGORITHM CODING
.
.
.
METHOD
-----
.
.
.
THIS ASSEMBLER SUBROUTINE USES EXECUTIVE REQUEST TSWAPS TO
SWAP REELS FOR A MULTI-VOLUME SET OF TAPES. THE CALLING PROGRAM
MUST PROVIDE A PROPERLY INITIALIZED PACKET AS FOLLOWS:
    INTEGER PACKET-NAME (3)
    PACKET-NAME (1) = 'FIRST 6-CHARS OF INTERNAL FILE NAME'
    PACKET-NAME (2) = 'NEXT 6 CHARS OF INTERNAL FILE NAME OR BLANKS'
    PACKET-NAME (3) = 'REEL NO OF NEXT TAPE' (FOR SWAP OPTION 2 ONLY)
THE FORTRAN CALLING SEQUENCE FOR THIS SUBROUTINE THEN IS AS FOLLOWS:
    CALL ERTSHP (LUPKT,IFUNC)
.
.
.
MACHINE-DEPENDENT CODE
-----
.
.
.
WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 6-BIT
FIELDATA CHARACTERS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.
BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.
DIFFERENT COMPILERS (EG., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.
.
.
.
EXTERNAL REFERENCES
-----
.
.
.
TSWAPS      0 EXECUTIVE REQUEST ROUTINE TO CLOSE REEL AND LOAD
            0 REQUESTED TAPE
.
.
.
EXCEPTIONS
-----
.
.
.
NONE
.
.
.
GLOBAL DECLARATIONS

```

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERTSMP/DAN  
002

```

. -----
.
.   NONE
.
.
. LOCAL DECLARATIONS
. -----
.
.   NONE
.
.
. PROCEDURE
. -----
.
.

```

S(00)  
ERTSMP

AXRS  
LMJ

LA  
LXI  
ER  
J  
END

A0.ATRACE  
'ERTSMP'  
A0.0.X11  
A0.1.X11  
TSHAPS  
3.X11

```

. BANKS REVERSED TO ELIMINATE INDIRECT LOAD
. TRACE
.
. XM = PACKET ADDRESS
. XI = FUNCTION
.
. RETURN

```

ERTHAT/DAN  
001

ORIGINAL PAGE IS  
OF POOR QUALITY

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

ERWAIT/DAN  
001

```

.      SUBROUTINE ERWAIT( 8 WAIT FOR COMPLETION OF PREVIOUSLY INITIATED I/O
.      U LUPKT)          8 PACKET (MUST BE PACKET USED TO INITIATE THE I/O)
.      -----
.
.      (E H SCHLOSSER)
.
.      THIS ASSEMBLER SUBROUTINE MAKES THE I/O FACILITIES OF ER WAITS DIRECTLY AVAIL-
.      ABLE TO A FORTRAN PROGRAM.
.
.      • SYNCHRONIZATION BUGS IN THE UNIVAC-SUPPLIED TAPE I/O HANDLER WITHIN THE
.      • EXEC CAUSE PREMATURE SETTING OF THE I/O STATUS TO COMPLETE, RESULTING
.      • IN SPORADIC TAPE MALFUNCTIONS WHEN THE SYSTEM IS LIGHTLY LOADED. THE
.      • IOWAIT CONSTRUCT ALLOWS THE USER TO SPECIFY (FOR TAPE I/O ONLY) A TIMED
.      • WAIT OF 0 THRU 63 MILLISECONDS TO BE TAKEN BY SUBROUTINE ERWAIT WHEN THE
.      • STATUS INDICATES I/O IS ALREADY COMPLETE ON ENTRY TO ERWAIT. IOWAIT
.      • SHOULD NORMALLY BE SET TO 10 MILLISECONDS WHEN NO OTHER EXEC REQUEST
.      • INTERVENES BETWEEN THE CALL TO ERIO AND THE CALL TO ERWAIT.
.
.
S(01)      AXRS
ERWAIT*     L,U      A2.0.X11      . ADDRESS OF PACKET
           LA,S1     A1.5.A2      . MILLISECONDS TO WAIT
           JZ        A1.00      . JUMP IF TIMED WAIT NOT REQUESTED
           TN        3.A2      . SKIP NI IF WAITING FOR I/O COMPLETION
           ER        THAITS     . TIMED WAIT FOR A1 MILLISECONDS
           TP        3.A2      . SKIP NI IF I/O ALREADY COMPLETE
           ER        WAITS      . WAIT FOR COMPLETION
           J         2.X11     . RETURN
           END
00

```

**DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS**

**SYPOET/DAN  
001**

SUBROUTINE SYPOET( 8 OUTPUT PROMPT IMAGE & GET INPUT IMAGE  
1 STAT. 8 STATUS OF GET: ' ' NORMAL COMPLETION  
'FS' FILE SEPARATOR (SEOF)  
'EOA' END OF ADD  
'EOF' END OF FILE (8...)  
'OFL' BUFFER OVERFLOW ( LENGTH > 120 )  
0 INOGET. 8 BUFFER FOR GET IMAGE (MAX 120 CHARS)  
0 LENOET. 8 LENGTH OBTEN IN CHARS. INCLUDES TRAILING BLANKS IF PRESENT  
1 LINPUT. 8 NUMBER OF LINES TO SPACE BEFORE PUTTING  
1 INOPUT. 8 BUFFER WITH PUT IMAGE  
1 LENPUT) 8 LENGTH TO PUT IN CHARS. INCLUDES TRAILING BLANKS IF PRESENT  
-----

**HISTORY**  
-----

E H SCHLOSSER LEMSCO 08/13/80 ORIGINAL CODE

**METHOD**  
-----

PROMPT, THEN READ NEXT SYSIN IMAGE. CONVERT EXEC-8 STATUS BITS & LENGTH  
IN UNIVAC WORDS TO STATUS STRING & LENGTH IN CHARACTERS.  
DECREMENT MADLEY UPON ENCOUNTERING THE END OF AN ADD-ED FILE.

**MACHINE-DEPENDENT CODE**  
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 8-BIT  
FIELDATA CHARACTERS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT,  
BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.  
DIFFERENT COMPILERS (EQ., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

**EXTERNAL REFERENCES**  
-----

ER TREADS 8 EXEC REQUEST: PROMPT, THEN READ FROM INPUT RUNSTREAM

**EXCEPTIONS**  
-----

1. IF THE GET BUFFER IS LESS THAN 120 CHARACTERS. RESULTS ARE UNDEFINED  
AND THE CURRENTLY EXECUTING PROGRAM MAY ABORT.

**GLOBAL DECLARATIONS**  
-----

AXRS

. STANDARD UNIVAC 1100 REGISTER MNEMONICS

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

SYPOET/DAN  
002

KONXQT

. COMMON PROGRAM EXECUTION SWITCHES

. LOCAL DECLARATIONS

\$(00) . D-BANK

PKT RES 2

. UNIVAC EXEC-8 TREADS PACKET

. PROCEDURE

\$(01) . I-BANK

SYPOET

. SET UP OUTPUT (PUT) SPECIFICATIONS

LA	A0.*3.X11	. # LINES TO SPACE BEFORE PUTTING
SZ	A1	. DIVISION COMING
LSSL	A0.6	. SHIFT # LINES 6 BITS TO LEFT
LA	A2.*5.X11	. LENGTH IN CHARS TO PUT
AA.U	A2.5	. LAST CHARACTER, COUNTING FROM 6
DI.U	A1.6	. A1 := LAST WORD, COUNTING FROM 1
		. A2 := LAST CHAR IN LAST WD. FROM 0
LA	A3.4.X11	. ADDR OF PUT IMAGE BUFFER
SA.M1	A0.PKT+0	. PACK
SA.S3	A1.PKT+0	. INTO
SA.M2	A3.PKT+0	. PKT+0

. MASK UNUSED CHARACTERS IN LAST WORD OF PUT BUFFER

LNA.U	A0.0	. DEMAND I/O STOP WORD
MS1.U	A2.6	. A2 := 1ST BIT OF LAST CHAR. FROM 0
SSL	A0.6.A2	. POSITION STOP CHARS OVER UNUSED BITS
ANA.U	A1.1	. LAST WORD, COUNTING FROM: 0
AA	A3.A1	. A3 := ADDR OF LAST WORD
LA	A4.A3	. SAVE ADDR OF LAST WORD
LA	A5.0.A3	. SAVE CONTENTS OF LAST WORD
OR	A0.0.A3	. MASK UNUSED CHARACTERS ...
SA	A1.0.A3	. ... IN LAST WORD

. SET UP INPUT (GET) SPECIFICATIONS

LA	A0.1.X11	. XM := ADDR OF GET IMAGE BUFFER
LXI	A0.(NO)	. XI := JUMP ADDR IF NO IMAGE TO GET
SA	A0.PKT+1	. PUT IN PKT+1

. REQUEST THE I/O

LA	A0.(PKT)	. REQUEST EXEC TO
ER	TREADS	. PUT & GET IMAGES

. INTERPRET THE I/O RESULTS

OK	LXI.XU	A0.0	. MASK STATUS BITS. LEAVING LENGTH IN MDS
	LA	A1.(	. STATUS IS OK
	TO.XU	A0.20	. SKIP N1 IF 20 > # MDS IN IMAGE
	LA	A1.(OFL	. IF A0 > 20, STATUS IS OVERFLOW
	MS1.XU	A0.6	. CONVERT LENGTH TO CHARACTERS
	J	RETURN	

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

SYNSET/DAN  
003

NO	LA	A2.A0	. SAVE STATUS BITS
	SZ	A0	. LENGTH IS 0
	JN	A2.EOF	. EOF IF BIT 35 IS ON
	LSSL	A2.2	. SHIFT BIT 33 INTO SIGN BIT
	JN	A2.EOA	. EOA IF BIT 33 IS ON
FS	LA	A1.(FS	. STATUS IS FILE SEPARATOR (EOF)
	J	RETURN	.
EOA	LA	A1.(EOA	. STATUS IS END OF ADD
	LA	A3.HADLEV	. CURRENT NESTED ADD LEVEL
	AA.XU	A3.-1	. ... LESS ONE
	JN	A3.RETURN	. CAN'T MAKE NEGATIVE
	SA	A3.HADLEV	. UPDATE ADD LEVEL
	J	RETURN	.
EOF	LA	A1.(EOF	. STATUS IS END OF FILE (8...)
RETURN		. RESTORE PUT BUFFER & RETURN WITH INTERPRETED I/O RESULTS	
	LA	A3.A4	. RESTORE ADDR OF LAST WORD IN BUFFER
	SA	A5.0.A3	. RESTORE CONTENTS OF LAST WORD
	SA	A0.2.X11	. STORE LENGTH OF GOTTEN IMAGE
	SA	A1.0.X11	. STORE STATUS
	J	7.X11	.
	END		

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

SYSADD/DAN  
881

```

SUBROUTINE SYSADD( 8 ADD DISK SYMBOLIC FILE OR ELEMENT TO SYSIN RUNSTREAM
0 LOCFIL. 8 LOCATION WITHIN FILE:      <0 NOT FOUND
C      "                                0 FILE
      1 NAMFIL. 8 NAME OF FILE          >0 ELEMENT LOCATION
      1 NAMELT. 8 NAME OF SYMBOLIC ELEMENT
      1 NAMVER) 8 NAME OF VERSION
C      NAMES MUST CONTAIN 12 CHARACTERS, INCLUDING TRAILING BLANKS
C      -----
C
C HISTORY
C -----
C      E M SCHLOSSER      LEC      01/17/79      ORIGINAL CODE
C
C METHOD
C -----
C      CHECK IF DISK SYMBOLIC FILE OR ELEMENT EXISTS. IF SO, INCREMENT MADLEV
C      (8ADD LEVEL COUNTER). CONSTRUCT 8 SUBMIT EXEC-8 8ADD.E IMAGE. (SYSET
C      DECREASEMENTS MADLEV UPON ENCOUNTERING THE END-OF-FILE GENERATED BY THE
C      8ADD.E AND RETURNS AN END-OF-ADD ('EOA') STATUS.)
C
C MACHINE-DEPENDENT CODE
C -----
C
C      UNIVAC EXEC 8 PROGRAM FILE NAMING CONVENTIONS AND EXECUTIVE REQUESTS.
C
C EXTERNAL REFERENCES
C -----
C
C      INTEGER LENCST      8 LENGTH OF CHARACTER STRING
C      MOVCSF      8 MOVE CHARACTER STRING
C      ERCSF      8 SUBMIT EXEC-8 FACILITY REQUEST CONTROL STATEMENT
C
C EXCEPTIONS
C -----
C
C      1. FILE OR ELEMENT DOES NOT EXIST.
C
C      2. FILE OR ELEMENT IS NOT SYMBOLIC.
C
C      3. FILE NOT ASSIGNED TO CURRENT RUN.
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE KOMXQT.LIST
C
C LOCAL DECLARATIONS

```



DAM PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

SYSADD/DAM  
002

```

C -----
C
      INTEGER NAMFIL(2),NAMELT(2),NAMVER(2)      8 ARGUMENTS
      INTEGER IMAGE(8)      8 ADD CONTROL STATEMENT IMAGE
      DATA (IMAGE(N),N=1,2)/'8ADD.E '/
      INTEGER LOCFIL      8 LOCATION WITHIN DSF
      INTEGER KSTART      8 START LOCATION WITHIN CHARACTER STRING OF SUBSTRING
      INTEGER LENGTH      8 LENGTH OF CHARACTER SUBSTRING

C
C
C PROCEDURE
C -----
C
C      CALL TRACE

C
C
C CHECK IF DSF EXISTS
C
      LOCFIL=LOCDSF(NAMFIL,NAMELT,NAMVER)
      IF(LOCFIL.LT.0) GO TO 900      8 NO SUCH DSF

C
C
C APPEND FILE NAME TO IMAGE
C
      KSTART=0      8 AFTER '8ADD.E '
      LENGTH=LENCST(NAMFIL,12)
      CALL MOVCST(IMAGE,(KSTART),(LENGTH), NAMFIL.(1),(LENGTH),' ')
      KSTART=KSTART+LENGTH
      CALL PUTCHR(IMAGE,(KSTART), ' ')
      KSTART=KSTART+1
      IF(LOCFIL.EQ.0) GO TO 800      8 FILE

C
C
C APPEND ELEMENT NAME TO IMAGE
C
      LENGTH=LENCST(NAMELT,12)
      CALL MOVCST(IMAGE,(KSTART),(LENGTH), NAMELT.(1),(LENGTH),' ')
      KSTART=KSTART+LENGTH
      IF(NAMVER(1).EQ.' ') GO TO 800      8 NO VERSION NAME

C
C
C APPEND VERSION NAME TO IMAGE
C
      CALL PUTCHR(IMAGE,(KSTART), ' ')
      KSTART=KSTART+1
      LENGTH=LENCST(NAMVER,12)
      CALL MOVCST(IMAGE,(KSTART),(LENGTH), NAMVER.(1),(LENGTH),' ')
      KSTART=KSTART+LENGTH

C
C
C SUBMIT 8ADD IMAGE TO EXEC
C
      800 CALL MOVCST(IMAGE,(KSTART),(4), ' ',(1),(4),' ')
      NADLEV=MAX0(NADLEV+1,1)      8 INCREMENT NESTED 8ADD LEVEL
      CALL ERCSF(1STAT, IMAGE)

```

**DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS**

**SYSADD/DAN  
003**

**C  
C**

**000 RETURN  
END**

```

SUBROUTINE SYSOET( 0 GET NEXT RECORD FROM SYSIN RUNSTREAM
0 INSTAT, 0 INPUT STATUS:      .      NORMAL COMPLETION
                                'FS'   FILE SEPARATOR (SEOF)
                                'EOA'   END OF ADD
                                'EOF'   END OF FILE (0...)
                                'OFL'   BUFFER OVERFLOW ( LENGTH > 120 )
0 IMAGE, 0 IMAGE BUFFER (MAX 120 CHARS)
0 LENGTH) 0 LENGTH OF IMAGE IN CHARS. INCLUDES TRAILING BLANKS. IF PRESENT
-----

```

# HISTORY

-----

```

E H SCHLOSSER   LEC   01/22/79   ORIGINAL CODE
E H SCHLOSSER   LEC   01/07/80   KOMXQT MACRO

```

# METHOD

-----

```

READ NEXT SYSIN IMAGE. CONVERT EXEC-0 STATUS BITS & LENGTH IN UNIVAC
WORDS TO STATUS STRING & LENGTH IN CHARACTERS.
DECREMENT MADLEY UPON ENCOUNTERING THE END-OF-FILE GENERATED BY THE 0ADD.E

```

# MACHINE-DEPENDENT CODE

-----

```

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 6-BIT
FIELDATA CHARACTERS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT,
BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES,
DIFFERENT COMPILERS (EO.. UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

```

# EXTERNAL REFERENCES

-----

```

ER READS      0 EXEC REQUEST: OBTAIN NEXT CARD IMAGE FROM INPUT RUNSTREAM

```

# EXCEPTIONS

-----

```

1. 7777

```

# GLOBAL DECLARATIONS

-----

```

AXRS
KOMXQT      . COMMON PROGRAM EXECUTION SWITCHES

```

# LOCAL DECLARATIONS

DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS

SYSOET/DAN  
802

-----

NONE.

PROCEDURE

SYSOET . 1-BANK

LA	A0.1.X11	. XM = IMAGE BUFFER ADDRESS
LXI	A0.(NO)	. XI = JMP ADDR IF NO IMAGE
ER	READS	. READ CARD IMAGE INTO BUFFER

OK	LXI.XU	A0.0	. MASK STATUS BITS. LEAVING LENGTH IN NOS
	LA	A1.1	. STATUS IS OK
	TO.XU	A0.20	. SKIP N1 IF 20 >= NOS IN IMAGE
	LA	A1.1'OFI	. IF A0 > 20. STATUS IS OVERFLOW
	MSI.XU	A0.8	. CONVERT LENGTH TO CHARACTERS
	J	RETURN	

NO	LA	A2.A0	. SAVE STATUS BITS
	SZ	A0	. LENGTH IS 0
	JN	A2.EOF	. EOF IF BIT 35 IS ON
	LSSL	A2.2	. SHIFT BIT 33 INTO SIGN BIT
	JN	A2.EOA	. EOA IF BIT 33 IS ON

FS	LA	A1.1'FS	. STATUS IS FILE SEPARATOR (SEOF)
	J	RETURN	

EOA	LA	A1.1'EOA	. STATUS IS END OF ADD
	LA	A3.NADLEV	. CURRENT NESTED ADD LEVEL
	AA.XU	A3.-1	. ... LESS ONE
	JN	A3.RETURN	. CAN'T MAKE NEGATIVE
	SA	A3.NADLEV	. UPDATE ADD LEVEL
	J	RETURN	

EOF	LA	A1.1'EOF	. STATUS IS END OF FILE (S...)
-----	----	----------	--------------------------------

RETURN	SA	A0.2.X11	. STORE LENGTH
	SA	A1.0.X11	. STORE STATUS
	J	4.X11	
	END		

**DAN PACKAGE APPENDIX P  
EXECUTIVE REQUESTS**

**STSPUT/DAN  
001**

**(NOT IMPLEMENTED)**

**PREFACE TO APPENDIX Q  
-----**

**THIS APPENDIX CONTAINS THE LIBRARY OF LABELLED COMMONS BLOCKS, GLOBAL MNEMONICS, PSEUDO RECORD STRUCTURES, AND COMMON STATEMENT FUNCTIONS FOR THE DAN PACKAGE. WHENEVER ANY CHANGE IS MADE IN THE PROCS WITHIN THIS APPENDIX, ALL PROGRAMS AND ROUTINES IN THE DAN PACKAGE MUST BE RE-COMPILED.**

**EACH PROGRAM, SUBROUTINE, OR FUNCTION WHICH REFERENCES A LABELLED COMMON BLOCK MUST 'INCLUDE.LIST' THE LIBRARY DEFINITION OF THE COMMON BLOCK FROM THIS APPENDIX. INLINE CODE EXPLICITLY DEFINING A LABELLED COMMON BLOCK MUST NEVER APPEAR WITHIN ANY PROGRAM, SUBROUTINE, OR FUNCTION!**

**DAN PACKAGE APPENDIX G  
MACROS**

**PREFACE-G  
002**

SPRT.SC DAM.PREFACE-Q	. (7912)	SET TABS 8 12 & 31
SPRT.SC DAM.APPENDIX-Q	.	
SHSO.N .ADJL4C	. ADJUSTED LINE FOR CORRECTED (SEE TRFORM-PROCS)	
SHSO.N .ADJS4C	. ADJUSTED SAMPLE FOR CORRECTED (SEE TRFORM-PROCS)	
SPRT.SC DAM.ALTPRT-PROCS	. COMMON/DEFINE FOR ALTERNATE PRINT FILES	
SPRT.SC DAM.ASHDEF-PROC	. DEFINE UNIVAC ASSEMBLER PARTIAL WDS IN FORTRAN V	
SHSO.N .ASHM1 ... ASHM2	. PARTIAL HALF-WORD MNEMONICS (SEE ASHDEF-PROC)	
SHSO.N .ASHS1 ... ASHS6	. PARTIAL SIXTH-WORD MNEMONICS (SEE ASHDEF-PROC)	
SHSO.N .ASHT1 ... ASHT3	. PARTIAL THIRD-WORD MNEMONICS (SEE ASHDEF-PROC)	
SHSO.N .AXRS	. STANDARD 1100 REG MNEMONICS (UNIVAC SYSTEM PROC)	
SPRT.SC DAM.CBDEF-PROC	. DEFINE CHARACTER BUFFER STRUCTURE & STD CB'S	
SHSO.N .CORL4A	. CORRECTED LINE FOR ADJUSTED (SEE TRFORM-PROCS)	
SHSO.N .CORS4A	. CORRECTED SAMPLE FOR ADJUSTED (SEE TRFORM-PROCS)	
SHSO.N .CORL4P	. CORRECTED LINE FOR PPD (SEE TRFORM-PROCS)	
SHSO.N .CORS4P	. CORRECTED SAMPLE FOR PPD (SEE TRFORM-PROCS)	
SHSO.N .CORL4S	. CORRECTED LINE FOR STM (SEE TRFORM-PROCS)	
SHSO.N .CORS4S	. CORRECTED SAMPLE FOR STM (SEE TRFORM-PROCS)	
SHSO.N .DIGITS	. (SEE FORPROCS)	
SPRT.SC DAM.FACBIT-PROC	. MNEMONICS FOR EXEC-8 CSFS FACILITY STATUS BITS	
SPRT.SC DAM.FIDEF-PROC	. DEFINE STRUCTURE OF FILE DEFINITION RECORD	
SHSO.N .FLDEF	. (SEE FORPROCS)	
SPRT.SC DAM.FORPROCS	. MISCELLANEOUS DEFINE PROCEDURES	
SPRT.SC DAM.GETOPT-APROC	. ASSEMBLER MANIPULATION OF XQT OPTION BITS/LETTER	
SHSO.N .ICBUF1	. STANDARD CHARACTER BUFFER #1 (SEE CBDEF-PROC)	
SHSO.N .ICBUF2	. STANDARD CHARACTER BUFFER #2 (SEE CBDEF-PROC)	
SHSO.N .ICBUF3	. STANDARD CHARACTER BUFFER #3 (SEE CBDEF-PROC)	
SHSO.N .ICBUF4	. STANDARD CHARACTER BUFFER #4 (SEE CBDEF-PROC)	
SHSO.N .LOADDR	. I/O ADDRESS OF BUFFER (SEE KOMIO-PROC)	
SHSO.N .IOAFCT	. I/O ABNORMAL FRAME COUNT (SEE KOMIO-PROC)	
SHSO.N .ILOCODE	. I/O STATUS CODE MNEMONIC (SEE KOMIO-PROC)	
SHSO.N .IOFUNC	. I/O FUNCTION (SEE KOMIO-PROC)	
SHSO.N .IONWDS	. I/O NUMBER OF WDS TRANSMITTED (SEE KOMIO-PROC)	
SHSO.N .IOSECT	. I/O SECTOR IN FILE (SEE KOMIO-PROC)	
SHSO.N .IOSIZE	. I/O BUFFER SIZE (SEE KOMIO-PROC)	
SHSO.N .IOSTAT	. I/O STATUS NUMBER (SEE KOMIO-PROC)	
SHSO.N .IOWAIT	. I/O WAIT SPEC (SEE KOMIO-PROC)	
SHSO.N .IOWORD	. I/O WORD IN FILE (SEE KOMIO-PROC)	
SHSO.N .KOMALT	. ALTERNATE PRINT FILE COUNTERS (SEE ALTPRT-PROCS)	
SHSO.N .KOMDET	. DETECTION FILE WINDOW PACKETS (SEE NERDET-PROCS)	
SHSO.N .KOMFIT	. ADJUSTMENT/REGISTRATION PARAMS (SEE NERDET-PROCS)	
SPRT.SC DAM.KOMIO-PROC	. FORTRAN MANIPULATION OF ASSEMBLER I/O PACKETS	
SPRT.SC DAM.KOMIRT-PROC	. COMMON IRRADIANCE TRANSFORMATION COEFFICIENTS	
SHSO.N .KOMIHW	. COMMON INPUT WINDOW PACKETS (SEE WINDOW-PROCS)	
SHSO.N .KOMKLS	. COMMON CLASSIFICATION INFO (SEE NERDET-PROCS)	
SPRT.SC DAM.KOMKS-PROC	. COMMON COLOR SCREEN PARAMETERS	
SHSO.N .KOMLOO	. LOO FILE I/O PKTS, POINTERS (SEE XQTLOO-PROCS)	
SPRT.SC DAM.KOMLU3-PROC	. COMMON I/O PACKET/POINTERS FOR UNIT 3	
SPRT.SC DAM.KOMLUS-PROC	. COMMON BUFFER, POINTERS, FLAGS FOR UNIT 5	
SPRT.SC DAM.KOML2N-PROC	. COMMON I/O PKTS FOR DETECTION FILES (UNITS 21-24)	
SHSO.N .KOMNER	. COMMON ERTS SCENE PARAMETERS (SEE NERDET-PROCS)	
SPRT.SC DAM.KOMNET-PROC	. COMMON CONTROL NETWORK COORDINATES	
SHSO.N .KOMOHM	. COMMON OUTPUT WINDOW PACKETS (SEE WINDOW-PROCS)	
SPRT.SC DAM.KOMSLM-PROC	. COMMON SPECTRAL LIMITS	
SPRT.SC DAM.KOMSYM-PROC	. COMMON SYMBOL TABLE	
SPRT.SC DAM.KOMTBL-PROC	. COMMON MULTI-PURPOSE TABLE	
SHSO.N .KOMXQT	. COMMON PROGRAM EXECUTION INFO (SEE XQTLOO-PROCS)	



DAM PACKAGE APPENDIX Q  
MACROS

APPENDIX-Q  
682

SPRT.SC DAM.KONXQT-APROC  
 SPRT.SC DAM.LSTLUB-PROC  
 SPRT.SC DAM.MAXBYT-PROC  
 SPRT.SC DAM.MAXICE-PROC  
 SPRT.SC DAM.MAXINT-PROC  
 SPRT.SC DAM.HERDET-PROCS  
 SHSO.N .NITAB  
 SHSO.N .NITROT  
 SPRT.SC DAM.NULCHR-PROC  
 SPRT.SC DAM.NULCST-PROC  
 SPRT.SC DAM.PICDEF-PROC  
 SHSO.N .PPOL4C  
 SHSO.N .PPDC4C  
 SPRT.SC DAM.PRCDEF-PROC  
 SPRT.SC DAM.PRODEF-PROC  
 SPRT.SC DAM.PXBDEF-PROC  
 SHSO.N .STMN4C  
 SHSO.N .STHE4C  
 SHSO.N .SYSXQT  
 SPRT.SC DAM.TRFORM-PROCS  
 SHSO.N .WINDEF  
 SPRT.SC DAM.WINDOW-PROCS  
 SHSO.N .XQTDEF  
 SPRT.SC DAM.XQTLOG-PROCS

- . COMMON PROGRAM EXECUTION INFO (ASH ROUTINES ONLY)
- . NAMELIST SPECS FOR UNIT 8 (REGISTRATN PARAMETERS)
- . DEFINE MAXIMUM BYTE VALUE
- . DEFINE MAXIMUM INTEGER-CHAR-EQUIV VALUE
- . DEFINE MAXIMUM INTEGER VALUE
- . COMMON/HEADER BLOCKS FOR ERTS DETECTION FILES
- . (SEE ALTPRT-PROCS)
- . (SEE ALTPRT-PROCS)
- . DEFINE NULL CHARACTER
- . DEFINE NULL CHARACTER STRING
- . DEFINE PICTAB PARAMETERS
- . PPD LINE FOR CORRECTED (SEE TRFORM-PROCS)
- . PPD COLUMN FOR CORRECTED (SEE TRFORM-PROCS)
- . DEFINE PRTCLASS PARAMETERS
- . DEFINE PRTDET PARAMETERS
- . DEFINE STRUCTURE OF PIXEL BUFFER
- . STM NORTHING FOR CORRECTED (SEE TRFORM-PROCS)
- . STM EASTING FOR CORRECTED (SEE TRFORM-PROCS)
- . (SEE XQTLOG-PROCS)
- . DEFINE COORDINATE TRANSFORMATIONS
- . (SEE WINDOW-PROCS)
- . COMMON/DEFINE FOR WINDOW PACKETS
- . (SEE XQTLOG-PROCS)
- . COMMON BLOCKS FOR SXQT & DAM LOG MANIPULATION

C ALTPRT-PROCS 8 ALTERNATE PRINT FILE (SPOOL FILE) COMMON / UNIT ALLOCATION

C -----

C

C

C HISTORY

C -----

C

C E M SCHLOSSER

LEC

03/03/74

ORIGINAL CODE

C E M SCHLOSSER

LEC

12/15/79

CHANGE COMMON NAME FROM ALT

C

C

C KOHALT PROC

C -----

C\*

COMMON/KOHALT/

8 ALTERNATE PRINT FILE POINTERS:

1 KOHALT(1).

8 NAME OF COMMON BLOCK

2 NCOPY.

8 NUMBER OF COPIES OF EACH FILE TO PRINT

3 NIT,NITMAX.

8 PRESENT, MAXIMUM MAP UNIT (NIT) IN THE CURRENT MAP

5 NITLO,NITHI.

8 LOW, HIGH MAP UNIT (NIT) IN THE CURRENT MAP SECTION

7 NITSLO,NITSHI

8 COUNTERS FOR ALLOCATING MAP UNITS (NITS) TO I/O UNITS

DATA KOHALT(1)='KOHALT'/

C\*

END

NITAB PROC

C -----

C\*

C\*

C\*

C\*

C\*

C\*

C\*

C\*

DEFINE NTAB(NIT)=10\*MOD((NIT+NITSLO),MALTN)

C\*

END

NITROT PROC

C -----

C\*

C\*

C\*

C\*

C\*

C\*

C\*

C\*

C\*

NITSLO=NITSHI-NITLO+1

NITSHI=NITSLO+NITHI

C\*

END

**DAN PACKAGE APPENDIX Q**  
**NACROS**

**ALTPRT-PROCS**  
**002**

```
C      ASHDEF-PROC  & DEFINE FORTRAN V EQUIV OF 1100 ASH PARTIAL WORD MNEMONICS
C      -----
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      07/07/74      ORIGINAL CODE IN FORPROCS
C      E H SCHLOSSER      LEC      12/15/79      SEPARATE INTO ASHDEF-PROC
C
C
C ASHDEF  PROC
C -----
C*
C*      THIS DEFINE PROCEDURE MAKES THE UNIVAC 1100 ASSEMBLER PARTIAL WORD
C*      MNEMONICS AVAILABLE TO A FORTRAN V PROGRAM.
C*
C*      DEFINE ASMS1(WORD)=FLD(00.6.WORD)
C*      DEFINE ASMS2(WORD)=FLD(06.6.WORD)
C*      DEFINE ASMS3(WORD)=FLD(12.6.WORD)
C*      DEFINE ASMS4(WORD)=FLD(18.6.WORD)
C*      DEFINE ASMS5(WORD)=FLD(24.6.WORD)
C*      DEFINE ASMS6(WORD)=FLD(30.6.WORD)
C*
C*      DEFINE ASMT1(WORD)=FLD(00.12.WORD)
C*      DEFINE ASMT2(WORD)=FLD(12.12.WORD)
C*      DEFINE ASMT3(WORD)=FLD(24.12.WORD)
C*
C*      DEFINE ASMH1(WORD)=FLD(00.18.WORD)
C*      DEFINE ASMH2(WORD)=FLD(18.18.WORD)
C*
C*      END
```

```

C      CDEF-PROC  & DEFINE CHARACTER BUFFER STRUCTURE & STANDARD CHAR BUFFERS
C      -----
C
C      C HISTORY
C      -----
C      E H SCHLOSSER      LEC      10/01/79      ORIGINAL CODE WITH CBSZIN=25
C
C      C METHOD
C      -----
C      CDEF DEFINES THE STRUCTURE OF CHARACTER BUFFERS USED BY THE CB4...
C      SERIES OF TRANSFORMS TO SUPPORT PASCAL-LIKE STREAM I/O.
C
C      ICBUF1 ... ICBUF5 ALLOCATE LOCATE CHARACTER BUFFER(S) WITHIN
C      EACH SEPARATELY-COMPILED MODULE IN WHICH THEY ARE INCLUDED.
C
C      C EXCEPTIONS
C      -----
C      1. AFTER ANY CHANGE TO THE CODE IN THIS PROC. ALL PROGRAMS, SUBROUTINES.
C      AND FUNCTIONS MUST BE RE-COMPILED.
C
C      2. THE INDIVIDUAL CHARACTER BUFFER ARRAYS MUST ALWAYS HAVE DIMENSION
C      SPECIFICATIONS EXACTLY EQUAL TO THE VALUE OF THE PARAMETER CBSZIN.
C
C      CDEF      PROC
C      -----
C*
C      PARAMETER CBSZIN=25      & CHARACTER BUFFER SIZE (INCL POINTERS) IN INTEGERS
C      PARAMETER CBNUL=CBSZIN-2  & HD IN CB CONTAINING NULCST
C      PARAMETER CBLOC=CBSZIN-1  & HD IN CB CONTAINING LOC OF LAST ACTUAL CHR
C      PARAMETER CBEND=CBSZIN-0  & HD IN CB CONTAINING LOC OF LAST POSSIBLE CHR
C*
C      END

ICBUF1  PROC
C-----
C      INTEGER ICBUF1(25)      & STANDARD CHAR BUFFER OF SIZE CBSZIN
C      END

ICBUF2  PROC
C-----
C      INTEGER ICBUF2(25)      & STANDARD CHAR BUFFER OF SIZE CBSZIN
C      END

```

DAN PACKAGE APPENDIX Q  
MACROS

CDEF-PROC  
002

ICBUF3 PROC  
C-----  
INTEGER ICBUF3(25) 3 STANDARD CHAR BUFFER OF SIZE CBSZIN  
END

ICBUF4 PROC  
C-----  
INTEGER ICBUF4(25) 3 STANDARD CHAR BUFFER OF SIZE CBSZIN  
END

ICBUF5 PROC  
C-----  
INTEGER ICBUF5(25) 3 STANDARD CHAR BUFFER OF SIZE CBSZIN  
END

```

C      FACBIT-PROC  & MNEMONICS FOR EXEC-8 CSFS FACILITY STATUS BITS
C      -----
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      07/07/74      ORIGINAL CODE IN FORPROCS
C      E M SCHLOSSER      LEC      12/19/79      SEPARATE INTO FACBIT-PROC
C
C
C FACBIT  PROC
C -----
C*
C*      THE DEFINE PROCEDURES IN THIS FORTRAN PROC EXTRACT COMMONLY USED STATUS
C*      BITS FROM THE FACILITY REQUEST (CSFS) STATUS WORD.
C*
C*      DEFINE REJECT(N)=FLO((00000).1.N)      & REJECTED
C*      DEFINE ACCEPT(N)=REJECT(COMPL(N))      & ACCEPTED
C
C*      DEFINE ALREADY(N)=FLO((135-33).1.N)      & ALREADY ASSIGNED/FREED
C*      DEFINE  NEWLY(N)=ALREADY(COMPL(N))      & NEWLY ASSIGNED/FREED
C*      DEFINE PRVCAT(N)=FLO((135-32).1.N)      & PREVIOUSLY CATALOGED
C*      DEFINE NOTCAT(N)=FLO((135-21).1.N)      & NOT CATALOGED
C*      DEFINE ROLOUT(N)=FLO((135-19).1.N)      & ROLLED OUT
C*      DEFINE WAITNO(N)=FLO((135-18).1.N)      & WAITING ON FACILITY
C*      DEFINE EXCLUS(N)=FLO((135-16).1.N)      & IN EXCLUSIVE USE BY ANOTHER RUN
C*      DEFINE OTHRUN(N)=FLO((135-15).1.N)      & ASSIGNED TO ANOTHER RUN
C*
C*      END

```

```

C      FIDEF-PROC  & DEFINE STRUCTURE OF FILE DEFINITION RECORD
C      -----
C
C      C HISTORY
C      -----
C      C A HELMKE      LEC      10/10/70      REQUIREMENTS
C      C A HELMKE      LEC      11/02/70      DESIGN & CODE
C
C      FIDEF  PROC
C      -----
C*
C*      PARAMETER
C*      MNEMONIC      MEANING
C*      -----
C*
C*      & FIDEQT=1.      & EQUIPMENT TYPE: 'TAPE'/'DISK'/'OTHR' (UNSUPPORTED
C*                      & DEVICE)/'NUL'/'ERR' (FILE NAME SYNTAX ERROR)
C*      & FIDEQC=2.      & EQUIPMENT CODE MNEMONIC
C*      & FIDTRK=3.      & NUMBER OF TRACKS: '7'/'9'/' '
C*      & FIDFPI=4.      & FLUX CHANGES PER INCH: '200'/'556'/'800'/'1600'/'
C*                      & '3200'/'???'
C*      & FIDBFH=5.      & BUFFER FORMAT: 'BST'/'BB'/'CST'/'ERR'
C*      & FIDCRL=6.      & CURRENT REEL NUMBER: 'XXXXXX'
C*      & FIDNRL=7.      & NEXT REEL NUMBER: 'XXXXXX'
C*      & FIDNOS=8.      & NOISE CONSTANT (INTEGER)
C*
C*
C*      NOTES
C*
C*      1.  FOR EQUIPMENT CODE MNEMONIC DEFINITIONS, SEE UNIVAC
C*          EXEC 8 HARDWARE/SOFTWARE SUMMARY FOR SASO SUBFIELD TYPE.
C*
C*      2.  IF FIDEQT <> 'TAPE', FIDTRK, FIDFPI, FIDCRL, FIDNRL,
C*          FIDBFH, AND FIDNOS ARE NOT APPLICABLE AND ARE BLANK
C*          (ZERO FOR FIDNOS)
C*
C*      3.  FIDBFH SPECIFIES THE FORMAT OF THE DATA IN THE BUFFER.
C*
C*      END

```



DIGITS PROC

```

C* (CHS)
C* THE DEFINE PROCEDURES IN THIS FORTRAN PROC EXTRACT THE ABSOLUTE
C* VALUE OF THE NAMED DECIMAL DIGIT FROM A BINARY INTEGER AND RETURN
C* THE DIGIT AS A FIELDATA CHARACTER. LEFT-JUSTIFIED IN A BLANK-FILLED
C* WORD.
C*
C*   DEFINE JONES(I) = IABS(I-10 * (1/10 )) * 2**30**0
C*   DEFINE JTENS(I) = (IABS(I-100 * (1/100 ))/10 ) * 2**30**0
C*   DEFINE JHUNS(I) = (IABS(I-1000 * (1/1000 ))/100 ) * 2**30**0
C*   DEFINE JTHOUS(I) = (IABS(I-10000 * (1/10000 ))/1000 ) * 2**30**0
C*
C*
C*   END

```

FLDEF PROC

```

C* (E H SCHLOSSER)
C* THE DEFINE PROCEDURES IN THIS FORTRAN PROC MANIPULATE 8-BIT CHARACTERS,
C* 8-BIT BYTES, AND 32-BIT (10M) WORDS, PACKED IN AN ARRAY OF 36-BIT (UNIVAC)
C* WORDS. ALL CHARACTERS/BYTES/WORDS ARE NUMBERED FROM THE LEFT OF THE
C* SPECIFIED ARRAY, STARTING WITH 1. ALL CHARACTERS/BYTES/WORDS ARE HANDLED
C* AS UNSIGNED BINARY, RIGHT-ALIGNED.
C*   DIMENSION UNIRAY(1)      8 DUMMY ARRAY FOR FORMAL ARGUMENT
C*   DEFINE ELMENT(UNIRAY,SUB)=UNIRAY(SUB)      8 FIX FOR COMPILER BUG
C*
C*
C*   THIS DEFINE PROCEDURE STORES/EXTRACTS CHARACTER NUMBER CNR INTO/FROM THE
C*   ARRAY SPECIFIED. RANGE: 0<CNR
C*   DEFINE CHAR(UNIRAY,CNR)
C*   8 =FLD(8*MOD(CNR+5,8),
C*   8      8,
C*   8      ELMENT(UNIRAY,(CNR+5)/8))
C*
C*
C*   THIS DEFINE PROCEDURE EXTRACTS (ONLY) THE FIELD STARTING AT CHARACTER
C*   NUMBER CSTART AND CHIDE CHARACTERS IN SIZE FROM THE ARRAY SPECIFIED.
C*   RANGE: 0<CSTART, 0<CHIDE<7
C*   DEFINE CHRFLD(UNIRAY,CSTART,CHIDE)      8 NOT YET IMPLEMENTED
C*
C*
C*   THIS DEFINE PROCEDURE EXTRACTS (ONLY) THE FIELD STARTING AT BYTE NUMBER
C*   BSTART AND BUIDE BYTES IN SIZE FROM THE ARRAY SPECIFIED.
C*   RANGE: 0<BSTART, 0<BUIDE<5
C*   NOTE: ALSO AVAILABLE AS EXTERNAL FUNCTION
C*   DEFINE BYTFLD(UNIRAY,BSTART,BUIDE)
C*   8 =FLD(MOD(8*BSTART+20,36),
C*   8      MIN(36-MOD(8*BSTART+20,36),8*BUIDE),
C*   8      ELMENT(UNIRAY,(8*BSTART+20)/36))
C*   8 *2**(8*BUIDE-MIN(36-MOD(8*BSTART+20,36),8*BUIDE))
C*   8 =FLD(8,
C*   8      8*BUIDE-MIN(36-MOD(8*BSTART+20,36),8*BUIDE),
C*   8      ELMENT(UNIRAY,(8*BSTART+20)/36+1))
C*   8 *MIN(8*BUIDE-MIN(36-MOD(8*BSTART+20,36),8*BUIDE),1)
C*
C*

```

**DAN PACKAGE APPENDIX Q  
MACROS**

**FORPROC9  
002**

C\*  
C\* THIS DEFINE PROCEDURE EXTRACTS (ONLY) 32-BIT WORD NUMBER MNR FROM THE  
C\* ARRAY SPECIFIED.  
C\* NOTE: ALSO AVAILABLE AS EXTERNAL FUNCTION  
C\* DEFINE (MNRD(UNIRAY,MNR)  
C\* & -BYTFLO(UNIRAY,MNR\*4-3.4)  
C\*  
C\* END

DAN PACKAGE APPENDIX Q  
MACROS

GETOPT-APROC  
001

```

.      GETOPT-APROC  & GET NEXT OPTION LETTERS FROM BITS IN REGISTER A5
.      -----
.
.  HISTORY
.  -----
.
.      E H SCHLOSSER      LEC      11/03/78      ORIGINAL CODE
.      E H SCHLOSSER      LEC      12/15/78      UPGRADE DOCUMENTATION
.
.  GETOPT-  PROC      .  CONVERT OPT BITS IN A5 INTO OPT LETTERS IN A2.A3.A4
.  -----
.
INIT      LA          A2.BLANKS      .
          LA          A3.BLANKS      .
          LA          A4.BLANKS      .
          LA          A1.SIEQZ       .  FIRST OPTION LETTER IS 'Z'
          J           NEWBIT
.
BLANKS
SIEQZ
SIEQ01
SAVREQ    RES      1
.
NEWBIT    SZ        A6              .  CLEAR LAST OPTION BIT
          DSL        A5.1           .  SHIFT NEW OPTION BIT INTO A6
          JZ        A6.NEHLTR       .  JUMP IF OPTION BIT NOT SET
          DSL        A3.6           .  SHIFT OPTION
          LSSL       A3.6           .  LETTERS (3 WORDS)
          DSL        A2.6           .  ONE LETTER TO RIGHT
          SA        A2.SAVREQ        .  PUT CURRENT OPTION LETTER
          OR        A1.SAVREQ        .  INTO S1 OF A2
.
NEHLTR    AN        A1.SIEQ01       .  DECREMENT OPTION LETTER
          JNZ       A1.NEWBIT       .  REPEAT FOR NEXT OPTION BIT. IF ANY
END

```

```

C      KONIO-PROC  & FORTRAN MANIPULATION OF EXEC-8 ASSEMBLER I/O PACKETS
C      -----
C
C      C HISTORY
C      -----
C
C      E H SCHLOSSER      LEC      08/27/74      ORIGINAL CODE W/ COMMON STATUS MSOS
C      C A MELNKE        LEC      11/11/78      ADD MNEMONIC STATUS CODES & DEL MSOS
C
C      KONIO      PROC
C      -----
C*
C*      INTEGER IONEHO(12) // ' ' 'EOF' 'EOF' 'EOF' ' ' 'EOF' ' ' ' ' ' '
C*      I/O ERROR CODE      00      01      02      03      04      05      06      07
C*      1                    'EOF' 'BADR' 'LOST' 'BADF'
C*      10      11      12      13
C*
C*      STORE I/O ARGUMENTS IN PACKET:
C*      INTEGER PKT(1)      & DUMMY PACKET
C*      DEFINE IOSIZE(PKT)=FLO(00.10.PKT(5)) & BUFFER SIZE
C*      DEFINE IOADDR(PKT)=FLO(10.10.PKT(5)) & BUFFER ADDRESS
C*      DEFINE IOWAIT(PKT)=FLO(00.00.PKT(6)) & MSEC TO WAIT - FIX FOR EXEC TP BUG
C*      DEFINE IOSECT(PKT)=PKT(6) & FILE-RELATIVE SECTOR ADDRESS (FASTRAND)
C*      DEFINE IOWORD(PKT)=PKT(6) & FILE-RELATIVE WORD ADDRESS (DRUM)
C*      DEFINE IOFUNC(PKT)=PKT(4) & FUNCTION (MUST BE SET BEFORE EACH CALL)
C*
C*      EXTRACT INFORMATION ON COMPLETED I/O FROM PACKET:
C*      DEFINE IOSTAT(PKT)=IOFUNC(PKT)/2**30 & EXEC-8 TYPE 1 STATUS CODE
C*      DEFINE IOAFCT(PKT)=FLO(12.00.PKT(4)) & ABNORMAL FRAME COUNT (TAPE)
C*      DEFINE IONHDS(PKT)=FLO(10.10.PKT(4)) & NUMBER OF WORDS TRANSMITTED
C*      DEFINE ERRXXX(K)=IONEHO(K+1) & DON'T REFERENCE EXPLICITLY
C*      DEFINE IOCODE(PKT)=ERRXXX(MING(IOFUNC(PKT)/2**30,1)) & TRANSLATION
C*      OF EXEC-8 TYPE 1 (I/O) STATUS CODES TO MNEMONICS:
C*      ' ' - NORMAL COMPLETION
C*      'EOF' - END OF FILE OR OUTSIDE OF FILE
C*      'BADR' - BAD I/O STATUS OF RECORD
C*      'BADF' - BAD I/O STATUS OF FILE
C*      'LOST' - LOST POSITION
C*
C      END

```

```

C      KOHIRT-PROC  8 COMMON IRRADIANCE TRANSFORMATION COEFFICIENTS
C      -----
C
C      HISTORY
C      -----
C
C      E H SCHLOSSER      LEC      12/19/78      ORIGINAL CODE IN ERTS-PROCS
C      E H SCHLOSSER      LEC      12/19/79      SEPARATE INTO KOHIRT-PROC
C
C      KOHIRT  PROC
C      -----
C*
      COMMON/KOHIRT/  8 IRRADIANCE TRANSFORMATION TYPE/WEIGHTS/GAINS/BIASES:
      1 KOHIRT(1).      8 NAME OF COMMON BLOCK
      2 IRTTYP.          8 TRANSFORMATION TYPE: 'RAW'.'LIN'('EAR'). 'POL'('AR)
      3 RTLWGT(5,2).    8 LINEAR TRANSF WEIGHTS (5 RAW CHANS. 2 TRANSFORMED CHANS)
      3 RTLOAN(2).       8 LINEAR TRANSF GAINS
      5 LRTW12(5,2).    8 LINEAR RADIANCE TRANSF WEIGHTED GAINS*2**12
      5 LRTB12(2).      8 LINEAR RADIANCE TRANSF BIASES*2**12
      7 NRTG12(2).      8 POLAR RADIANCE TRANSF GAINS*2**12
      9 NRTB24(2).      8 POLAR RADIANCE TRANSF BIASES*2**24
      1 :RTF12(5,2)    8 SHARPENING FILTER COEFFICIENTS*2**12
      DATA KOHIRT(1) / 'KOHIRT' /
C*
      END

```

C KONKS-PROC & COMMON COLOR SCREEN PARAMETERS

C -----

C  
C HISTORY  
C -----

C E H SCHLOSSER LEMSCO 02/22/80 ORIGINAL CODE

C  
C MACHINE-DEPENDENT CODE  
C -----

- C  
C 1. USE OF 912 TO SHIFT BY ONE ASCII BYTE .S DICTATED BY UNIVAC 1100  
C INTERNAL BYTE STORAGE.  
C  
C 2. THE VALUES OF KSOFF AND KSON ARE THOSE WHICH CAUSE THE INFOTON-100  
C TERMINAL TO SWITCH OUTPUT BETWEEN ITS OWN B&W ALPHANUMERIC SCREEN  
C AND THE ISC INTECOLOR CRT SCREEN CONNECTED TO ITS PRINTER PORT.  
C  
C 3. THE VALUE OF KSCLER IS THAT WHICH CLEARS THE INTECOLOR SCREEN AND SETS  
C THE FOREGROUND COLOR TO WHITE.  
C  
C 4. THE KSKIKE TABLE CONVERTS DAM INTEGER-COLOR-EQUIVALENTS (IKE'S) TO  
C THE COLOR CONTROL CODES (KIKE'S) USED BY THE INTECOLOR. NOTE THAT THE  
C INTECOLOR DOES NOT HAVE COLORS CORRESPONDING TO ODD-NUMBERED IKE'S.  
C HENCE EACH INTECOLOR CODE IS ASSIGNED TO TWO IKE'S.  
C

C KONKS PROC

C -----

C\*

COMMON/KONKS/ & COLOR SCREEN PARAMETERS:  
1 KONKS(1). & NAME OF COMMON BLOCK  
2 KSLINE. & COLOR SCREEN NUMBER OF LINES  
3 KSCOLM. & COLOR SCREEN NUMBER OF COLUMNS  
4 KSKIKE(15) & COLOR SCREEN CODES ASSIGNED TO IKE'S  
DATA KONKS(1)/'KONKS'/  
DATA KSLINE/45/  
DATA KSCOLM/79/

C

C DATA KSKIKE/  
C KIKE COLOR IKE COLOR  
8 20. 8 BLUE 00 BLUE  
8 20. 8 BLUE 01 CYAN-BLUE  
8 22. 8 CYAN 02 CYAN  
8 22. 8 CYAN 03 AQUA  
8 18. 8 GREEN 04 GREEN  
8 18. 8 GREEN 05 YELLOW-GREEN  
8 19. 8 YELLOW 06 YELLOW  
8 19. 8 YELLOW 07 ORANGE  
8 17. 8 RED 08 RED  
8 17. 8 RED 09 RED-MAGENTA  
8 21. 8 MAGENTA 10 MAGENTA  
8 21. 8 MAGENTA 11 (UNASSIGNED)  
8 16. 8 BLACK 12 BLACK

**DAN PACKAGE APPENDIX Q  
MACROS**

**KONKS-PROC  
002**

```

      8 18. 8 BLACK      13 (UNASSIGNED)
      8 23/ 8 WHITE     14 WHITE
C
      PARAMETER KSOFF = 027*512 + 110      8 COLOR SCREEN OFF
C      ASCII:         <ESC> + LOWERCASE 'N'
      PARAMETER KSON  = 027*512 + 078      8 COLOR SCREEN ON
C      ASCII:         <ESC> + UPPERCASE 'N'
      PARAMETER KSLER = 012*512 + 023      8 CLEAR SCREEN & COLOR=WHITE
C      ASCII:         <FF> + CTRL-N
C*
END

```

```

C      KOMLU3-PROC  & COMMON I/O PACKET/POINTERS FOR UNIT 3
C      -----
C
C      HISTORY
C      -----
C
C      E H SCHLOSSER      LEC      12/27/73      ORIGINAL CODE IN FORPROCS
C      E H SCHLOSSER      LEC      12/01/75      SEPARATE INTO IO-PROCS
C      HELMKE/TOMPKINS    LEC      10/11/78      SPLIT INTO KOMLU3-PROC & REORGANIZE
C      E H SCHLOSSER      LEMSCO   06/20/80      REPLACE LU310Q WITH LU3LRS
C
C      KOMLU3  PROC
C      -----
C*
C*      PARAMETER V3MAX = 5      & MAXIMUM # OF TAPE VOLUMES SUPPORTED
C*
C*      COMMON/KOMLU3/      & I/O AND UNPACKING DATA FOR MSS/RBV DATA (UNIT 3):
C*      1 LU3LRS.          & LENGTH OF DISK RECORD (IN UNIVAC SECTORS)
C*      2 LU3PKT(8).       & I/O PACKET
C*      3 LU3FID(10).      & FILE ID INFO IN FIDEF FORMAT
C*      4 LU3BIL.          & NUMBER OF PHYSICAL TAPE BLOCKS PER SCAN LINE
C*      5 LU3CBR.          & NUMBER OF CONSECUTIVE BAD REC (INITIALIZE TO 0)
C*      6 LU3MBR.          & MAXIMUM NUMBER OF 'BADR' BEFORE 'BADR'-'BADF'
C*      7 LU3MIB,LU3BIB.   & NUMBER OF WORDS,BYTES (EXCL PADDING) PER PHYSICAL
C*                          TAPE BLOCK
C*      8 LU3BFM.          & BUFFER FORMAT:
C*                          'BST' - INTERNAL BYTE STRING
C*                          'BB' - 8-BIT BYTE STRING
C*                          'NUL' - NO BYTES
C*                          ANYTHING ELSE - UNSUPPORTED BYTE FORMAT
C*      9 LU3SEQ(2).       & (1) DATA SEQUENCE , (2) VERSION:
C*                          'NUL' - NO SEQUENCE
C*                          '0' - 1978
C*                          'BIP' - BAND INTERLEAVED BY PIXEL PAIR
C*                          '0' - 4 CHANNEL
C*                          '1' - 5 CHANNEL
C*                          'BIL' - BAND INTERLEAVED BY LINE
C*                          '0' - 1978
C*                          'BSQ' - BAND SEQUENTIAL
C*                          '0' - 1978
C*                          'DSK' - RANDOM ON DISK
C*                          '0' - 1978
C*      10 LU3REF(2).      & (1) RECORD FORMAT , (2) VERSION
C*                          'NUL' - NO RECORD
C*                          '0' - 1978
C*                          'AM' - MSS UNCORRECTED FROM MDP
C*                          '0' - 1978
C*                          'PM' - MSS FULLY CORRECTED FROM MDP
C*                          '0' - 1978
C*                          'AR' - RBV UNCORRECTED FROM MDP
C*                          '0' - 1978
C*                          'PR' - RBV FULLY CORRECTED FROM MDP
C*                          '0' - 1978
C*                          'ERT' - MSS UNCORRECTED (ORIGINAL ERTS FORMAT)

```



**KOML U3-PROC**  
**002**

**Q-20**

DAN PACKAGE APPENDIX Q  
MACROS

KOHLUS-PROC  
001

```

C      KOHLUS-PROC  & COMMON I/O BUFFERS/POINTERS FOR FREE-FORMAT INPUT (UNIT 5)
C      -----
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      12/27/73      ORIGINAL CODE IN FORPROCS
C      E H SCHLOSSER      LEC      12/01/75      SEPARATE INTO 10-PROCS
C      E H SCHLOSSER      LEC      01/29/79      SPLIT INTO KOHLUS-PROC & REORGANIZE
C
C
C KOHLUS  PROC
C -----
C*
      COMMON/KOHLUS/      & I/O BUFFERS/POINTERS FOR FREE-FORMAT INPUT (UNIT 5):
      1 LUSSPN.           & MAXIMUM REMAINING LINES TO SPAN
      2 LUSREQ.           & REQUIRED FIELD SWITCH: 0=OPTIONAL, 1=REQUIRED
      3 LUSTOS.           & I/O STATUS: ' ', 'FS', 'EOA', 'EOF', 'OFL'
      4 LUSIMO(20).       & IMAGE BUFFER (MAX 120 CHARACTERS)
      5 LUSLLH(3)         & IMAGE POINTERS:
                          DEFINE LUSLOC=LUSLLH(1) & LOCATION OF PREVIOUS/CURRENT DATA FIELD
                          DEFINE LUSLEN=LUSLLH(2) & LENGTH OF PREVIOUS/CURRENT DATA FIELD
                          DEFINE LUSMAX=LUSLLH(3) & LENGTH OF LUSIMO ( *(-1) IF END )
      INTEGER KOHLUS(1)
      EQUIVALENCE (LUSSPN,KOHLUS(1))
C*
END

```

DAM PACKAGE APPENDIX Q  
MACROS

KOHL2N-PROC  
001

```

C      KOHL2N-PROC  & COMMON I/O PACKETS FOR DETECTION FILES (UNITS 21 THRU 24)
C      -----
C
C
C      C HISTORY
C      -----
C
C      E M SCHLOSSER      LEC      12/27/73      ORIGINAL CODE IN FORPROCS
C      E M SCHLOSSER      LEC      12/01/78      SEPARATE INTO IO-PROCS
C      E M SCHLOSSER      LEC      12/06/79      SEPARATE INTO KOHL2N-PROC
C      E M SCHLOSSER      LEC      05/16/80      NO DATA INITIALIZATION OF UNIT NAMES
C
C
C      KOHL2N  PROC
C      -----
C*
      COMMON/KOHL2N/      & I/O PACKETS FOR DETECTION FILES (UNITS 21 THRU 24):
      1 L21PKT(8).      & I/O PACKET FOR BUFFER 1 OR UNIT 21 (FILE *DAMDET-1)
      2 L22PKT(8).      & I/O PACKET FOR BUFFER 2 OR UNIT 22 (FILE *DAMDET-2)
      3 L23PKT(8).      & I/O PACKET FOR BUFFER 3 OR UNIT 23 (FILE *DAMDET-3)
      4 L24PKT(8)      & I/O PACKET FOR BUFFER 4 OR UNIT 24 (FILE *DAMDET-4)
      INTEGER L2NPKT(8,4)
      EQUIVALENCE (L21PKT(1),L2NPKT(1,1))
C*
      END

```

```

C      KONNET-PROC  & COMMON CONTROL NETWORK POINT NUMBERS & COORDINATES
C      -----
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      03/29/73      ORIGINAL CODE IN COEF
C      E H SCHLOSSER      LEC      10/16/75      PUT IN LABELLED COMMON
C      E H SCHLOSSER      LEC      12/15/79      DELETE RAHNET COORDINATES
C
C
C KONNET  PROC
C -----
C*
      COMMON/KONNET/      & CONTROL NETWORK DATA:
      1 KONNET(1).        & NAME OF COMMON BLOCK
      2 NETH1.            & HIGHEST NODE IN NET
      3 NETPT(350).        & POINT NUMBER
      3 ADJNET(2,350).    & OSFC-ADJUSTED SCAN COORDINATES
      3 CORNET(2,350).    & CORRECTED SCAN COORDINATES (FROM MODEL)
      3 GEDNET(2,350).    & GEOGRAPHIC COORDINATES (DEGREES)
      3 STMNET(2,350).    & SCENE TRANSVERSE MERCATOR COORDINATES (METRES)
      3 CTDLIN,CTDSAM.    & NETWORK CENTROID (LINE, SAMPLE)
      5 CTDLAT,CTDLON    & NETWORK CENTROID (LATITUDE, LONGITUDE)
      DATA KONNET(1)/~KONNET*/
C*
END

```

DAN PACKAGE APPENDIX Q  
MACROS

KOMSLH-PROC  
001

C KOMSLH-PROC & COMMON SPECTRAL LIMITS

C -----

C HISTORY

C -----

C E H SCHLOSSER LEC 05/03/73 ORIGINAL CODE IN FORPROCS  
C E H SCHLOSSER LEC 12/15/79 SEPARATE INTO KOMSLH-PROC

C KOMSLH PROC

C -----

C\*

COMMON/KOMSLH/ & SPECTRAL LIMITS:  
& LINVAL(128.6) & 0 TO 127 FOR LIMCH(2), (3), (4) MINIMUM & MAXIMUM

C\*

END

```

C      KONSYN-PROC  & COMMON SYMBOL TABLE
C      -----
C
C      HISTORY
C      -----
C
C      E H SCHLOSSER      LEC      07/07/74      ORIGINAL CODE IN FORPROCS
C      E H SCHLOSSER      LEC      02/01/79      SEPARATE INTO KONSYN-PROC
C      E H SCHLOSSER      LEC      12/20/79      REPLACE KSYBIT WITH NCISYM
C      E H SCHLOSSER      LEMSCO   09/30/80      NOINFO/NO DATA I-I-E & I-K-E
C
C      MACHINE-DEPENDENT CODE
C      -----
C
C      ASSUMES AT LEAST 8 CHARACTERS PER KSYM ELEMENT.
C      ASSUMES UNIVAC 1100 FIELDATA CHARACTER SET.
C
C      KONSYN  PROC
C      -----
C*
C*      PARAMETER ISYHLO=0      & INFO LOW VALUE
C*      PARAMETER ISYHNI=255    & INFO HIGH VALUE
C*      PARAMETER ISYHNI=ISYHNI+1  & 'NO INFO' VALUE
C*      PARAMETER ISYHND=ISYHNI+2  & 'NO DATA' VALUE
C*      PARAMETER KSYMSZ=1+ISYHND  & SIZE OF SYMBOL TABLE IN WORDS
C*
C*      COMMON/KONSYN/
C*      1 NCISYM.      & MAXIMUM NUMBER OF CHARACTERS PER SYMBOL OF CURRENT SYMBOLS
C*      2 KSYM(KSYMSZ) & CHARACTER STRING SYMBOL TABLE:
C*      KSYM(1+ISYHLO)      & CHAR STRING SYMBOL FOR INFO LOW VALUE
C*      KSYM(1+ISYHNI)      & CHAR STRING SYMBOL FOR INFO HIGH VALUE
C*      DATA KSYM(1+ISYHNI) //  &1'// & CHAR STRING SYMBOL FOR 'NO INFO' VALUE:
C*                               STRING = ' '
C*                               I-I-E = 00 (INTENSITY = 10%)
C*                               I-K-E = 14 (COLOR = WHITE)
C*      DATA KSYM(1+ISYHND) //:  &1'// & CHAR STRING SYMBOL FOR 'NO DATA' VALUE:
C*                               STRING = ' '
C*                               I-I-E = 05 (INTENSITY = 50%)
C*                               I-K-E = 14 (COLOR = WHITE)
C*
C*      END

```

```

C      KONTBL-PROC  8 COMMON MULTI-PURPOSE TABLE
C      -----
C
C      C HISTORY
C      -----
C
C      E M SCHLOSSER      LEC      12/27/73      ORIGINAL CODE IN FORPROCS
C      E M SCHLOSSER      LEC      12/01/78      SEPARATE INTO KONTBL-PROCS
C      MELNKE/TOMPKINS     LEC      10/11/79      DELETE TAPE HEADER STUFF
C
C      KONTBL  PROC
C      -----
C*
C*      PARAMETER KTBLSZ=1000  8 NUMBER OF WORDS IN KTABLE ARRAY
C*      COMMON/KONTBL/
C*      1 KTBLTY.              8 TABLE TYPE: (' ', 'TICK', 'FREQ', 'FACT')
C*      2 KTBLNW.              8 TABLE WINDOW NUMBER
C*      3 KTABLE(KTBLSZ)      8 TABLE FOR TICKS, FREQUENCY, FACTORS
C*
C*      TICK TABLE, ONE WORD PER TICK PACKED AS FOLLOWS:
C*      DEFINE LINTIC(INTIC)=FLO(00.10.KTABLE(INTIC))  8 PRINT/PLOT LINE
C*      DEFINE COLTIC(INTIC)=FLO(10.17.KTABLE(INTIC))  8 PRINT/PLOT COLUMN
C*      DEFINE LEVTIC(INTIC)=FLO(35.01.KTABLE(INTIC))  8 TICK LEVEL:
C*      0 = PRIMARY TICK
C*      1 = SECONDARY TICK
C*
C*      INTEGER KFCHAN(8)      8 CHANNELS USED IN FREQUENCY TABLE
C*      INTEGER KFREQ(128,8)   8 FREQUENCY TABLE FOR DISPLAYS/MAPS
C*      INTEGER KFCRO(10,19)   8 CROSS FREQ TABLE (10 INT X 19 COLORS)
C*      EQUIVALENCE (KTABLE(1),KFCHAN)
C*      EQUIVALENCE (KTABLE(11),KFREQ)
C*      EQUIVALENCE (KTABLE(801),KFCRO)
C*
C*      ORTHOGONAL FACTOR INFO DERIVED FROM CORRELATION & ROTATION **
C*      INTEGER KPIXCO      8 NUMBER OF PIXELS CORRELATED
C*      INTEGER KCHACO      8 NUMBER OF CHANNELS CORRELATED
C*      INTEGER KFACCO      8 NUMBER OF FACTORS DERIVED FROM CORRELATION
C*      EQUIVALENCE
C*      8 (KTABLE(1),KPIXCO),
C*      8 (KTABLE(2),KCHACO),
C*      8 (KTABLE(3),KFACCO)
C*      REAL CMNEAN(7)      8 MEANS OF ORIGINAL CHANNELS
C*      REAL CMSTD(7)      8 STANDARD DEVIATIONS OF ORIGINAL CHANNELS
C*      REAL PCTVAR(2,7)    8 INDIVIDUAL/CUMULATIVE PERCENT OF VARIANCE
C*      EQUIVALENCE
C*      8 (KTABLE(10),CMNEAN),
C*      8 (KTABLE(20),CMSTD),
C*      8 (KTABLE(30),PCTVAR)
C*      REAL CORREL(7,7)    8 CORRELATION COEFFICIENTS OF ORIGINAL CHANNELS
C*      REAL EIOVAL(7,7)    8 EIGENVALUES ON DIAGONAL, ZEROS OFF DIAGONAL
C*      REAL EIOVEC(7,7)    8 EIGENVECTORS

```

**DAN PACKAGE APPENDIX Q  
MACROS**

**KONTBL-PROC  
002**

```
REAL FSTRUC(7,7)  & FACTOR STRUCTURE (CORR BET CHANNELS & FACTORS)
REAL FSTROT(7,7)  & ROTATED FACTOR STRUCTURE
REAL FCNORM(7,7)  & FACTOR COEFFICIENTS OF NORMALIZED CHANNELS
REAL FCNROT(7,7)  & ROTATED FACTOR COEFFICIENTS OF NORMALIZED CHANNELS
REAL FCORIO(7,7)  & FACTOR COEFFICIENTS OF ORIGINAL CHANNELS
REAL FCOROT(7,7)  & ROTATED FACTOR COEFFICIENTS OF ORIGINAL CHANNELS
EQUIVALENCE
& (KTABLE(90),CORREL),
& (KTABLE(100),EIOVAL),
& (KTABLE(150),EIOVEC),
& (KTABLE(200),FSTRUC),
& (KTABLE(250),FSTROT),
& (KTABLE(300),FCNORM),
& (KTABLE(350),FCNROT),
& (KTABLE(400),FCORIO),
& (KTABLE(450),FCOROT)
```

**C\*  
END**



KONXQT-APROC 8 ASH COMMON BLOCK FOR PROGRAM EXECUTION SWITCHES, COUNTERS  
-----

HISTORY  
-----

E M SCHLOSSER	LEC	12/01/78	ORIGINAL CODE IN INDIVIDUAL ROUTINES
E M SCHLOSSER	LEC	01/03/79	MADLEY COUNTER
E M SCHLOSSER	LEC	10/22/79	NCOLOR SWITCH
E M SCHLOSSER	LEC	01/07/80	SEPARATE INTO KONXQT-APROC

EXCEPTIONS  
-----

1. DECLARATIONS OF KONXQT COMMON VARIABLE NAMES IN THIS ASSEMBLER PROC MUST ALWAYS MATCH THOSE IN THE CORRESPONDING FORTRAN PROC.

KONXQT: PROC  
-----

\$(02) . FORTRAN LABELLED COMMON

KONXQT	INFO	2	2 . FORTRAN LABELLED COMMON
COMMON	RES	2	. REST OF COMMON ALLOCATED BY FORP
MBATCH:	EQUF	COMMON+0..\$1	. 1-BATCH. 0-DEMAND
MCIRM:	EQUF	COMMON+0..\$2	. 1-CONFIRM. 0-DON'T
MCHECK:	EQUF	COMMON+0..\$3	. 1-CHECKOUT. 0-PRODUCTION
MECHO:	EQUF	COMMON+0..\$4	. 1-ECHO. 0-DON'T
MDATA:	EQUF	COMMON+0..\$5	. 1-DATA/CHECKOUT
MADLEV:	EQUF	COMMON+0..\$6	. DYNAMIC CSFS BADD LEVEL
MLEND:	EQUF	COMMON+1..\$1	. 1-LEGEND. 0-NONE
NCOLOR:	EQUF	COMMON+1..\$2	. 1-COLORCRT. 0-NONE
			(INTERVENING SWITCH RESERVED FOR FUTURE USE)
MPROMT:	EQUF	COMMON+1..\$4	. 1-PROMPT. 0-DON'T
MDUMP:	EQUF	COMMON+1..\$5	. 1-DUMP. 0-DON'T
MTRACE:	EQUF	COMMON+1..\$6	. 1..03=TRACE.0-DON'T

END

```

C      LSTLUS-PROC  8 NAMELIST SPECIFICATIONS FOR UNIT 8 (REGISTRATION PARAMETERS)
C      -----
C
C
C      HISTORY
C      -----
C
C      E H SCHLOSSER      LEC      12/27/73      ORIGINAL CODE IN FORPROCS
C      E H SCHLOSSER      LEC      12/31/73      SEPARATE INTO LSTLUS-PROC
C      E H SCHLOSSER      LEC      12/15/78      ADD MERGED
C
C
C      METHOD
C      -----
C
C      THE NAMELIST STATEMENTS IN THIS FORTRAN PROC ARE USED FOR
C      STORING REGISTRATION PARAMETERS ON UNIT 8. LOADING EXACT REGISTRATION
C      PARAMETERS FROM UNIT 8. AND LOADING NOMINAL REGISTRATION PARAMETERS FROM
C      AN ELEMENT IN THE DAM FILE.
C
C
C      LSTLUS  PROC
C      -----
C
C      NAMELIST/LSTNER/      8 ER/S SCENE NUMBER:
C      1 NERTS(1),NERTS(2),NERTS(3)
C
C*
C      NAMELIST/LSTORB/      8 ORBITAL PARAMETERS:
C      2 NERLIN,NERSAN.      8 NUMBER OF LINES, SAMPLES
C      3 ALTKM,ALTSAN.      8 ALTITUDE IN KILOMETRES, SAMPLES
C      7 CTRLIN,CTRAN.      8 SCENE CENTER LINE, SAMPLE
C      8 CTRLAT,CTRLON.      8 SCENE CENTER LATITUDE, LONGITUDE (DEGREES)
C      9 DIRLAT,DIRLON.      8 SCENE NADIR LATITUDE, LONGITUDE (DEGREES)
C      0 PITDEC,ROLDEC.      8 SCENE PITCH, ROLL (DEGREES)
C      1 YANDEC.      8 SCENE YAW (DEGREES)
C      2 MERGED      8 GEOMETRY -- WILL APPEAR AS INTEGER:
C
C*      'ERT'      11129000013
C*      'HOM'      14200027420
C*      'LCC'      10300040003
C*      'PS'      22052503013
C*      'SOM'      20110007403
C*      'UTH'      20341457221
C
C*
C      NAMELIST/LSTSCN/      8 SCANNER PARAMETERS:
C      1 SCNA,SCNB,SCNC.
C      2 ROLRAC,SCNTH2,SCNTIM
C
C*
C      NAMELIST/LSTFIT/      8 LEAST SQUARES FIT PARAMETERS:
C      0 NCTLPT,PCTCTL.
C      0 RMSMET.
C      7 UTHCND,STNCND.
C      8 CONSTH(1),CONSTH(2),CONSTH(3),CONSTH(4),CONSTH(5),CONSTH(6).
C      9 STNCOR(1),STNCOR(2),STNCOR(3),STNCOR(4),STNCOR(5),STNCOR(6).
C      0 CONSRH(1),CONSRH(2),CONSRH(3),CONSRH(4),CONSRH(5),CONSRH(6)
C
C*
C      END

```

DAN PACKAGE APPENDIX Q  
MACROS

MAXBYT-PROC  
001

MAXBYT PROC

C\*

PARAMETER MAXBYT = 255 0 LARGEST INTEGER IN 8-BIT BYTE

C\*

END

**DAM PACKAGE APPENDIX Q  
MACROS**

**MAXICE-PROC  
001**

**MAXICE PROC**

**C\***

**PARAMETER MAXICE = 63    8 MAXIMUM INTEGER CHARACTER EQUIVALENT  
C FOR UNIVAC FORTRAN V**

**C\***

**END**

DAM PACKAGE APPENDIX Q  
MACROS

MAXINT-PROC  
001

MAXINT PROC

C\*  
C\* MAXIMUM SIGNED INTEGER WHICH WILL YIELD CORRECT RESULTS USING UNIVAC  
C\* FORTRAN V COMPILER. (A SIGNED INTEGER TWICE AS LARGE CAN ACTUALLY BE  
C\* REPRESENTED IN ONE UNIVAC 1100 SERIES COMPUTER WORD, BUT DUE TO  
C\* DESIGN FLAWS IN THE UNIVAC FORTRAN V COMPILER IT WILL NOT ALWAYS  
C\* YIELD CORRECT RESULTS.)

PARAMETER MAXINT = 17179869183 8 2\*\*34-1

C\*  
END

C NERDET-PROCS 8 COMMON/HEADER BLOCKS FOR ERTS DETECTION FILES

C  
C  
C  
C HISTORY  
C -----

C	E M SCHLOSSER	LEC	12/27/73	ORIGINAL CODE IN FORPROCS
C	E M SCHLOSSER	LEC	12/19/75	SPLIT INTO ERTS-PROCS & ADD KLSTYP
C	E M SCHLOSSER	LEC	07/01/78	ADD LCVTOL & KTIPIX
C	HELMKE/TOMPKINS	LEC	10/11/79	RENAME & REORGANIZE FOR MDP TAPES
C	E M SCHLOSSER	LEC	05/18/80	IN KOMKLS REPLACE NBFCHO WITH NOUTCH

C  
C KOMNER PROC  
C -----

C\* THIS LABELED COMMON IS PART (SECTORS 0-2) OF THE DETECTION FILE HEADER.  
C\* WORDS 1-8 UNIQUELY IDENTIFY AN ERTS SCENE.

C\* PARAMETER SIZNER = 64

C*	COMMON/KOMNER/	3 ERTS SCENE PARAMETERS:
C*	1 NERSAT(2).NERSEN.	3 SATELLITE. SENSOR
C*	4 NERTS(3).	3 ERTS-NO. DAYS-SINCE-LAUNCH. GMT
C*	7 NERLIN.NERSAM.	3 NUMBER OF LINES. SAMPLES OF DATA IN FULL SCENE (USED FOR REGISTRATION. EXCLUDES PAD PIXELS)
C*	9 NERDAY.NERMON.NERYR.	3 DAY(INTEGER). MONTH(ALPHA). YEAR(INTEGER)
C*	2 NERSEL.NERSAZ.	3 SUN ELEVATION. SUN AZIMUTH (DEGREES)
C*	4 NERCHA.	3 NUMBER OF CHANNELS ALLOCATED IN FILE (DATA IS NOT NECESSARILY PRESENT FOR ALL CHANNELS)
C*	5 ALTKM.ALTSAM.	3 ALTITUDE (KILOMETRES. SAMPLES)
C*	7 CTRLIN.CTRSAM.	3 SCENE CENTER (LINE. SAMPLE) (USED FOR REGISTRATION)
C*	9 CTRLAT.CTRLON.	3 SCENE CENTER (LATITUDE. LONGITUDE)
C*	1 DIRLAT.DIRLON.	3 SCENE NADIR (LATITUDE. LONGITUDE)
C*	3 PITDEG.ROLDEG.	3 PITCH. ROLL AT SCENE CENTER (DEGREES)
C*	5 YAHDEG.WHYDEG.	3 YAW. HEADING MINUS YAW AT SCENE CENTER (DEGREES)
C*	7 NERGE0.	3 GEOMETRY OF IMAGE DATA:

C*		'LCC' - LAMBERT CONFORMAL CONIC PROJECTION
C*		'PS' - POLAR STEREOGRAPHIC PROJECTION
C*		'SOM' - SPACE OBLIQUE MERCATOR PROJECTION
C*		'UTM' - UNIVERSAL TRANSVERSE MERCATOR PROJECTION
C*		'NOM' - HOTINE OBLIQUE MERCATOR PROJECTION
C*		'ERT' - ERTS (UNRESAMPLED)

C*	8 NERRES.	3 RESAMPLING ALGORITHM:
C*		'C' - CUBIC CONVOLUTION
C*		'N' - NEAREST NEIGHBOR
C*		'.' - NONE
C*	9 NERCOR.	3 TYPE OF CORRECTION APPLIED TO UNCLASSIFIED DATA:
C*		'U' - UNCORRECTED
C*		'S' - SYSTEM
C*		'G' - GEOMETRIC CONTROL POINTS
C*		'R' - RELATIVE (DUMMY) CONTROL POINTS

C\* 0 NERSAN(10).NERGAI(10).NERTHO(10). 3 FOR EACH CHANNEL:

**DAN PACKAGE APPENDIX Q  
MACROS**

**NERDET-PROCS  
002**

```

C*          BAND: '1'/'2'/'3'/'4'/'5'/'6'/'7'/'8'/'A'/'B'/'C'/'D'/' '
C*
C*          GAIN: 'H' - HIGH / 'L' - LOW / ' ' - NO DATA
C*
C*          TRANSMISSION MODE: '1' - LINEAR / '2' - COMPRESSED /
C*          ' ' - NO DATA
C*
C*          0 NERADN.          8 'A' - ASCENDING / 'D' - DESCENDING MODE
C*          1 NERPAT.NERROW.  8 NOMINAL PATH. ROW (CHARACTER STRING), ' ' IF UNDEFINED
C*          3 NCCT.NCCTOT     8 CCT STRIP #, TOTAL # CCT STRIP(S) FOR THIS SCENE
C*
C*          INTEGER KOMNER(SIZNER)
C*          EQUIVALENCE (NERSAT(1),KOMNER(1))
C*
C*          END

```

```

KOMKLS  PROC
C-----
C*
C*          THIS LABELED COMMON IS PART (SECTORS 3-4) OF THE DETECTION FILE HEADER.
C*          WORDS 1-28 SHOULD UNIQUELY CHARACTERIZE A SET OF SPECTRAL CLASSIFICATION
C*          LIMITS.
C*
C*          PARAMETER SIZKLS = 29
C*
C*          COMMON/KOMKLS/
C*          1 KLSTYP.          8 DISP/DET/MAP TYPE: 'RAD'/'GRA'/'LAP'/'DEN'/'CLA'/'OLD'/'NUL'
C*          2 NTERAL(4).       8 CLASS NAME (BINARY CLASSIFICATION ONLY)
C*          6 NOUTCH.          8 NUMBER OF OUTPUT CHANNELS
C*          7 NBFCHR(5).       8 NUMBER OF BUFFER CONTAINING RAW CHANNEL 1/2/3/4/5
C*          2 NLIMCH.          8 NUMBER OF LIMIT CHANNELS
C*          3 LIMCH(5).        8 LIMIT CHANNEL CONTAINED IN BUFFER NUMBER 1/2/3/4/5
C*          8 LCVLO(5).        8 LIMIT CHANNEL LOW VALUES
C*          3 LCVHI(5).        8 LIMIT CHANNEL HIGH VALUES
C*          8 LCVTOL.          8 'NEAR HIT' TOLERANCE FOR LIMIT CHANNEL VALUES
C*          8 KTIPIX          8 COUNTS PER PIXEL
C*
C*          INTEGER KOMKLS(SIZKLS)
C*          EQUIVALENCE (KLSTYP,KOMKLS(1))
C*
C*          EQUIVALENCE (LIMCH(1),LIMCH1),(LCVLO(1),LCVLO1),(LCVHI(1),LCVHI1)
C*
C*          END

```

```

KOMFIT  PROC
C-----
C*
C*          THIS LABELED COMMON IS PART (SECTORS 5-8) OF THE DETECTION FILE HEADER.
C*          WORDS 1-17 UNIQUELY CHARACTERIZE A CONTROL NETWORK.
C*
C*          PARAMETER SIZFIT = 42
C*

```

```

COMMON/KOMFIT/
1 SCNA,SCNB,SCNC.  3 MIRROR SCAN COEFFICIENTS (RADIAN)
4 ROLRAC.          3 ROLRAD*SCNC
5 SCNTH2.          3 HALF OF ACTIVE SCAN ANGLE (RADIAN)
6 SCNTIM.          3 RADIAN PER SAMPLE AT RADIAN NATURAL FREQUENCY
7 NCTLPT,PCTCTL.  3 NUMBER OF CONTROL POINTS & PERCENT OF SCENE COVERED
8 RMSHET.          3 ROOT-MEAN-SQUARE ERROR (METERS) OF ADJUSTMENT
9 UTHCHD.          3 UTH CENTRAL MERIDIAN (DEGREES)
10 STHCHD.         3 STH CENTRAL MERIDIAN (DEGREES)
11 CONSTH(8).      3 COEFFICIENTS -- CORRECTED TO SCENE TRANSVERSE MERCATOR
12 STHCOR(8).      3 COEFFICIENTS -- SCENE TRANSVERSE MERCATOR TO CORRECTED
13 CONSRH(8).      3 COEFFICIENTS -- CORRECTED TO SCENE ROTATED MERCATOR
14 IRFD.           3 REPRESENTATIVE FRACTION DENOMINATOR
15 CORPPD(8).      3 COEFFICIENTS -- CORRECTED TO PRINT/PLOT DEVICE
16 PPDOR(8).       3 COEFFICIENTS -- PRINT/PLOT DEVICE TO CORRECTED

C*
  INTEGER KOMFIT(SIZFIT)
  EQUIVALENCE (SCNA,KOMFIT(1))

C*
END

```

KOMDET PROC

```

C-----
C*
C* THIS LABELED COMMON IS PART (SECTORS 7-8) OF THE DETECTION FILE HEADER.
C* UNUSED JENMDY ELEMENTS MUST CONTAIN BLANKS. OTHER UNUSED VARIABLES IN
C* THIS COMMON BLOCK MUST CONTAIN BINARY ZEROS. VARIABLES IN THIS COMMON
C* MUST ONLY BE USED BY ROUTINES IN THE OPEN2N/READ2N/CLOSE2N SERIES.
C*
  PARAMETER SIZDET = 32

C*
  COMMON/KOMDET/  3 WINDOWS AND GENERATION DATES FOR 4 DETECTION FILES:
C*              AXIS      NODE      FILE (CCT)
1 MSADWH (2.      3.          4          1. 3 DETECTION WINDOW PACKETS
2 JENMDY(4).      3 GENERATION MONTH, DAY, YEAR FOR EACH DETECTION FILE
3 LOETRS(4)       3 LENGTH OF DETECTION FILE RECORDS (IN UNIVAC SECTORS)

C*
  DATA          3 PACKET SIZE (NODES):
1 MSADWH(2.1.1)/3/
2 MSADWH(2.1.2)/3/
3 MSADWH(2.1.3)/3/
4 MSADWH(2.1.4)/3/

C*
  INTEGER KOMDET(SIZDET)
  EQUIVALENCE (MSADWH(1.1.1),KOMDET(1))

C*
END

```



DAN PACKAGE APPENDIX Q  
MACROS

NULCHR-PROC  
001

NULCHR PROC  
C\*  
PARAMETER NULCHR=  
C\*  
END

DAN PACKAGE APPENDIX Q  
MACROS

NULCST-PROC  
001

NULCST PROC

C\*

PARAMETER NULCST=-0

8 NULL CHARACTER STRING (UNIVAC FORTRAN V)

C\*

END

**DAM PACKAGE APPENDIX Q  
MACROS**

**PICDEF-PROC  
001**

```
PICDEF  PROC
C*      (E H SCHLOSSER)
C*
C*      PARAMETER HALTHI=4      8 MAXIMUM HALTH FOR PICTAB
      PARAMETER KPAQHI=132     8 MAXIMUM KPAGE FOR PICTAB
C*
END
```

DAN PACKAGE APPENDIX Q  
MACROS

PRCDEF-PROC  
001

```
PRCDEF  PROC
C **
C ** (J C CRISP)
C **
C ** THIS PROC SETS MAXIMUM VALUES FOR THE NUMBER OF ALTERNATE
C ** PRINT FILES AND NUMBER OF COLUMNS PER PAGE TO BE USED IN
C ** PRTCLASS.
C **
C **
      PARAMETER
      & NALTHI=9.      & MAX # OF ALTERNATE PRINT FILES
      & KPAQHI=132    & MAX # OF COLUMNS PER PAGE
C **
C **
END
```

**DAN PACKAGE APPENDIX Q  
MACROS**

**PRODEF-PROC  
001**

**PRODEF PROC**

**C \*\* (E M SCHLOSSER)**

**C \*\***

**C \*\***

**PARAMETER NALTHI=4**

**8 MAXIMUM NALTH FOR PRIDEY**

**PARAMETER KPAOMI=132**

**8 MAXIMUM KPAGE FOR PRIDEY**

**C \*\***

**END**

DAM PACKAGE APPENDIX G  
MACROS

PXDEF-PROC  
001

```

C      PXDEF-PROC  & DEFINE STRUCTURE OF PIXEL BUFFER
C      -----
C
C      HISTORY
C      -----
C      E H SCHLOSSER      LEC      08/11/79      REQUIREMENTS
C      J C CRISP          LEC      08/02/79      DESIGN & CODE
C
C      PXDEF  PROC
C      -----
C*
C*      MNEMONICS FOR POINTERS TO LOCATIONS IN PIXEL BUFFER
C*      DEFINING BUFFER STRUCTURE
C*
C*
C*      PARAMETER
C*      MNEMONIC      MEANING
C*      -----
C*
C*      PREAMBLE PORTION OF BUFFER
C*
C*      & PXREC=1.      & RECORD NUMBER
C*      & PXLINO=2.     & LINE NUMBER
C*      & PXCHAN=3.     & CHANNEL NUMBER
C*      & PXQUAL=4.     & QUALITY CODE:
C*
C*      0 -- GOOD DATA
C*      1 -- DATA BASED ON SUBSTITUTED LINE
C*      2 -- FILLED LINE ON INPUT
C*      3 -- FILLED LINE ON OUTPUT
C*      4 -- QUALITY CODE UNAVAILABLE DUE TO TRUNCATION
C*      5 -- NO DATA
C*
C*      & PXBINT=5.     & BINTYPE: 'CHR'/'BYT'/'INT'/'NUL'(NOT APPLICABLE)
C*      & PXLBIN=6.     & LOWEST BIN CONTAINING DEFINED PIXEL (FIRST DEFINED
C*                        PIXEL). 0 IF NO DEFINED PIXELS
C*      & PXL5AM=7.     & LOWEST SAMPLE/COLUMN NUMBER REFERENCING DEFINED
C*      & PXLCOL=7.     & PIXEL (FIRST DEFINED PIXEL). 0 IF
C*                        NO DEFINED PIXELS
C*      & PXHBIN=8.     & HIGHEST BIN CONTAINING DEFINED PIXEL (LAST DEFINED
C*                        PIXEL). 0 IF NO DEFINED PIXELS
C*      & PXH5AM=9.     & HIGHEST SAMPLE/COLUMN NUMBER REFERENCING DEFINED
C*      & PXHCOL=9.     & PIXEL (LAST DEFINED PIXEL). 0 IF
C*                        NO DEFINED PIXELS
C*      & PXNOIN=10.    & 'NO INFO' FLAG
C*      & PXNOOA=11.    & 'NO DATA' THRESHOLD
C*      & PXLJOI=12.    & LOW JOIN (0 IF UNKNOWN OR NOT APPLICABLE)
C*      & PXHJOI=13.    & HIGH JOIN (0 IF UNKNOWN OR NOT APPLICABLE)
C*
C*      BIN PORTION OF BUFFER
C*
C*      & PXBINS=14     & BINS
C*
C*
C*

```

NOTES

1. PIXELS IN ALL CHANNELS ARE THE SAME IN SIZE, ACHIEVED IF NECESSARY BY RESAMPLING PIXELS FROM LOW RESOLUTION CHANNELS.
2. A LOW RESOLUTION CHANNEL WHICH IS NOT RESAMPLED IS TREATED AS HAVING NO DEFINED PIXELS.
3. PIXELS FOR THE SAME EARTH LOCATION FROM DIFFERENT CHANNELS ARE ALWAYS ASSIGNED THE SAME SAMPLE NUMBER.
4. SAMPLE NUMBER 1 CORRESPONDS TO THE LEFTMOST PIXEL IN THE SCENE DEFINED IN AT LEAST ONE HIGH RESOLUTION CHANNEL. PIXELS TO THE LEFT OF THIS ARE ALWAYS CONSIDERED UNDEFINED.
5. THE HIGHEST POSSIBLE SAMPLE NUMBER CORRESPONDS TO THE RIGHTMOST PIXEL IN THE SCENE FOR WHICH A RADIANCE VALUE IS DEFINED IN AT LEAST ONE HIGH RESOLUTION CHANNEL. PIXELS TO THE RIGHT OF THIS ARE ALWAYS CONSIDERED UNDEFINED.
6. THE INPUT WINDOW MUST ALWAYS BE A SINGLE ENVELOPE WITHOUT VERTICES.
7. WHEN THE OUTPUT IS MACHINE READABLE (IN ERTSOUP, CLASSIFY, ETC.) THE ENTIRE AREA INSIDE THE ENVELOPE OF THE OUTPUT WINDOW PACKET WILL BE PROCESSED AND THE VERTICES WILL BE IGNORED.
8. WHEN THE OUTPUT IS GRAPHIC (IN PICTAB, COLORTAB, PRYCLASS, ETC.) THE AREA INSIDE THE ENVELOPE BUT OUTSIDE THE VERTICES OF THE OUTPUT WINDOW PACKET WILL BE MASKED WITH NO-DATA THRESHOLD.
9. FILTERING WILL ALWAYS BE DONE BEFORE MASKING.

END

TRFORM PROC

```

C*      (END)
C*      THE DEFINE PROCEDURES IN THIS FORTRAN PROC TRANSFORM COORDINATES
C*      BETWEEN THE VARIOUS SYSTEMS USED BY LANDSAT AND THE DAN PACKAGE.
C*      THE ARGUMENT LISTS FOR THESE DEFINE PROCEDURES ARE DESIGNED TO PERMIT
C*      FUTURE REFINEMENTS IN THE MODELS. ARGUMENTS NOT CURRENTLY USED IN THE
C*      MODELS APPEAR WITHIN THE DEFINITIONS AS (ARG-ARG) OR (ARG*0). COMPILER
C*      OPTIMIZATION REMOVES THE ADDITIONAL CODE THUS GENERATED. IF THESE
C*      ARGUMENTS HAD NOT APPEARED WITHIN THE DEFINITIONS, THE COMPILER WOULD
C*      HAVE GENERATED SPURIOUS DIAGNOSTICS.
C*
C*      THE MODEL FOR CORRECTING ADJUSTED SCANNER COORDINATES INCLUDES:
C*      ACCELERATION IN MIRROR SCAN VELOCITY
C*      PANORAMIC EFFECT
C*      CONSTANT ROLL WITHIN A SCENE
C*
C*      CORRECTED FROM ADJUSTED:
C*
C*      DEFINE CORL4C(ADJLIN,ADJSAM)=ADJLIN*(ADJSAM-ADJSAM)      & ADJSAM NOT USED
C*      DEFINE COR54C(ADJLIN,ADJSAM)=CTRSAM*ALTSAM*TAN
C*      &      (ROLRAC-SCNTH2*SCNA*SIN
C*      &      (SCNTIM*(ADJSAM-1)*SCNB))*(ADJLIN-ADJLIN)      & ADJLIN NOT USED
C*
C*      ADJUSTED FROM CORRECTED:
C*
C*      DEFINE ADJL4C(CORLIN,CORSAM)=CORLIN*(CORSAM-CORSAM)      & CORSAM NOT USED
C*      DEFINE ADJS4C(CORLIN,CORSAM)=1.*(ASIN(AMAX1(-1.,AMIN1(1.,
C*      &      ((ATAN((CORSAM-CTRSAM)/ALTSAM)-ROLRAC-SCNTH2)/SCNA)))
C*      &      -SCNB)/SCNTIM*(CORLIN-CORLIN)      & CORLIN NOT USED
C*
C*      SCENE TRANSVERSE MERCATOR FROM CORRECTED:
C*
C*      DEFINE STNM4C(CORLIN,CORSAM)=
C*      &      CORSTH(1)*CORLIN-CORSTH(2)*CORSAM-CORSTH(3)
C*      DEFINE STNE4C(CORLIN,CORSAM)=
C*      &      CORSTH(4)*CORLIN-CORSTH(5)*CORSAM-CORSTH(6)
C*
C*      CORRECTED FROM SCENE TRANSVERSE MERCATOR:
C*
C*      DEFINE CORL4S(STNE,STNM)=
C*      &      STNCOR(1)*STNM-STNCOR(2)*STNE-STNCOR(3)
C*      DEFINE COR54S(STNE,STNM)=
C*      &      STNCOR(4)*STNM-STNCOR(5)*STNE-STNCOR(6)
C*
C*      PRINT/PLOT FROM CORRECTED:
C*
C*      DEFINE PPOL4C(CORLIN,CORSAM)=
C*      &      CORPPD(1)*CORLIN=      0.*CORSAM-CORPPD(3)      & CORPPD(2)=0.
C*      DEFINE PPOC4C(CORLIN,CORSAM)=

```



**DAM PACKAGE APPENDIX G  
MACROS**

**TRFORM-PROCS  
802**

```

      8          CORPPD(4)*CORLIN+CORPPD(5)*CONSAM+CORPPD(6)
C*
C*
C*      CORRECTED FROM PRINT/PLOT:
C*
      DEFINE CORL4P(PPOLIN,PPDCOL)=
      8          PPDCOR(1)*PPOLIN+      0.*PPDCOL+PPDCOR(3)      8 PPDCOR(2)*0.
      DEFINE COR5P(PPOLIN,PPDCOL)=
      8          PPDCOR(4)*PPOLIN+PPDCOR(5)*PPDCOL+PPDCOR(6)
C*
      END

```

```

C      WINDOW-PROCS  & DEFINE INPUT, OUTPUT WINDOW PACKETS & THEIR STRUCTURE
C      -----
C
C      C HISTORY
C      -----
C
C      E H SCHLOSSER      LEC      09/03/74      ORIGINAL CODE
C      E H SCHLOSSER      LEC      12/01/75      REPLACE WENTER BY WUSED
C      E H SCHLOSSER      LEC      12/19/79      REPLACE WINC BY WSP100, DELETE RAM
C
C      THE STANDARD PACKET USED IN THE DAN PACKAGE TO DEFINE A TWO-DIMENSIONAL
C      WINDOW IS A VARIABLE LENGTH ARRAY (EITHER INTEGER OR REAL) OF TWO-WORD
C      NODES IN THE FORM: WINPKT(AXIS.NODE).  THE HEAD NODE CONTAINS A POINTER
C      TO THE HIGHEST NODE CURRENTLY USED AND THE PACKET SIZE IN NODES.  THESE
C      TWO NUMBERS ARE ALWAYS STORED AS INTEGERS, REGARDLESS OF THE ARRAY TYPE.
C
C      WINDEF  PROC
C      -----
C*
C*      ALL AXIS AND NODE MNEMONICS ARE SHOWN BELOW:
C*
C*
C*      PARAMETER
C*      AXIS      AXIS
C*      MNEMONIC   MEANING
C*      -----
C*
C*      & WENTER=1.      & HIGHEST NODE USED (IN NODE WHEAD ONLY -- (OLD FORM))
C*      & WUSED=1.      & HIGHEST NODE USED (IN NODE WHEAD ONLY)
C*      & WSIZE=2.      & PACKET SIZE IN NODES (IN NODE WHEAD ONLY)
C*      & WLAT=1.      & LATITUDE
C*      & WLOX=2.      & LONGITUDE
C*      & WLIN=1.      & LINE
C*      & WSAH=2.      & SAMPLE
C*      & WCOL=2.      & COLUMN
C*      & WEA =1.      & EASTING
C*      & WNO =2.      & NORTHING
C*      & WX =1.      & X
C*      & WY =2.      & Y
C*
C*
C*      PARAMETER      USED      NODE
C*      USED/NODE      MEANING    CONTENTS
C*      MNEMONIC      -----
C*
C*      & WNUL =0.      & NO WINDOW
C*      & WHEAD=1.      &
C*      & WMIN =2.      &
C*      & WMAX =3.      &
C*      & WSP100=4.      & NO TICKS OR ORIGIN
C*      & WTIC =5.      &
C*      & WTIC=1
C*
C*      WINDOW HEAD
C*      ENVELOPE MINIMUM
C*      ENVELOPE MAXIMUM
C*      DATA INCREMENTS*100
C*      PRIMARY TICK INTERVALS
C*      SECONDARY TICK INTERVALS

```

DAN PACKAGE APPENDIX Q  
MACROS

WINDOW-PROCS  
002

	8	WORIO=7.	8	NO VERTICES	ORIGIN OF VERTICES
	8	MVER =8	8		CLOSING VERTEX
C*		MVER+1		RECTANGLE VECTOR	1ST VERTEX
C*		MVER+2		RECTANGLE DIAGONAL	2ND VERTEX
C*		MVER+3		TRIANGLE BOUNDARY	3RD VERTEX
C*		MVER+N		N-GRAM BOUNDARY	NTH VERTEX
C*					
C*					
C*					
C*		NOTES:			
C*					
C*		1.		POLYGRAM VERTICES ARE ALWAYS STORED IN COUNTER-CLOCKWISE (CCH) ORDER	
C*					
C*		2.		THE CLOSING VERTEX OF AN N-GRAM CONTAINS A DUPLICATE OF THE N-TH	
C*				VERTEX (LAST VERTEX).	
C*					
C*		END			

KOMIHW PROC

C-----

C\*  
 PARAMETER WHI=3      8 NUMBER OF NODES IN INPUT WINDOW PACKETS  
 COMMON/KOMIHW/      8 INPUT WINDOW PACKETS:  
 1 KOMIHW(1).      8 NAME OF COMMON BLOCK  
 2 HSAIHW(2,WHI).      8 MULTISPECTRAL SCAN ADJUSTED COORDINATES  
 3 CORIHW(2,WHI)      8 CORRECTED SCAN COORDINATES (FROM MODEL)  
 DATA KOMIHW(1)='KOMIHW'/  
 DATA      8 PACKET SIZE (NODES):  
 2 HSAIHW(2,1)/WHI/  
 3 CORIHW(2,1)/WHI/      8 INTEGER IN REAL ARRAY!  
 C\*  
 END

KOMOHW PROC

C-----

C\*  
 PARAMETER WHO=26      8 NUMBER OF NODES IN OUTPUT WINDOW PACKETS  
 COMMON/KOMOHW/      8 OUTPUT WINDOW PACKETS:  
 1 KOMOHW(1).      8 NAME OF COMMON BLOCK  
 0 KSYOHW(8).      8 OUTPUT COORDINATE SYSTEM NAMES FOR EACH NODE  
 2 HSAOHW(2,WHO).      8 MULTISPECTRAL SCAN ADJUSTED COORDINATES  
 3 GEOOHW(2,WHO).      8 GEOGRAPHIC COORDINATES (DEGREES)  
 4 UTHOHW(2,WHO).      8 UTM COORDINATES (METERS)  
 5 PPDOHW(2,WHO)      8 PRINT/PLOT DEVICE COORDINATES  
 DATA KOMOHW(1)='KOMOHW'/  
 DATA      8 PACKET SIZE (NODES):  
 2 HSAOHW(2,1)/WHO/  
 3 GEOOHW(2,1)/WHO/      8 INTEGER IN REAL ARRAY!  
 4 UTHOHW(2,1)/WHO/      8 INTEGER IN REAL ARRAY!  
 5 PPDOHW(2,1)/WHO/      8 INTEGER IN REAL ARRAY!  
 C\*  
 END

**DAN PACKAGE APPENDIX Q  
MACROS**

**WINDOM-PROCS  
003**

```

C      XQTLOG-PROCS  3 COMMON BLOCKS FOR 3XQT & DAM LOG FILE MANIPULATION
C      -----
C
C      C HISTORY
C      -----
C
C      E M SCHLOSSER      LEC      12/27/73      ORIGINAL CODE IN FORPROCS
C      E M SCHLOSSER      LEC      12/01/75      SEPARATE INTO XQTLOG-PROCS
C      E M SCHLOSSER      LEC      01/03/79      MADLEV COUNTER
C      E M SCHLOSSER      LEC      10/22/79      BIGGER LOG BUFR. HQQ PKT. MCOLR SM
C      E M SCHLOSSER      LEMSCO   08/25/80      LOCAL MODE SWITCH & MODE SWITCH SAVE
C

```

SYSXQT PROC

C-----

```

C*
C*      -- FOR INCLUSION IN BLOCK DATA SUBROUTINE SYS-BLOCK ONLY --
C*      -- MUST NOT BE INCLUDED IN EXECUTABLE SUBROUTINES !!!
C*

```

```

COMMON/KONXQT/
1 MSWTC(2).          3 PROGRAM ID AND STATUS INFO:
2 JMDY,JHMS,JOHM.    3 MODE SWITCHES (NO DEFINES IN BLOCK DATA SUB)
3 JPRTCN,JRUN(3),JPROG(4),JMDYHM(3).  3 PROGRAM 3XQT DATE/TIME (FIELDATA)
4 JHDO(12.2).        3 PROGRAM HEADING LINE 0
5 LINCH,KINCH.       3 HEADING LINES 1 & 2
6 LPAGE,KPAGE.       3 DAM LINES/INCH. COLUMNS/INCH
7 LSINCH.            3 DAM LINES/PAGE. COLUMNS/PAGE
8 LSPAGE.            3 INSTALLATION SYS-GEN LINES/INCH
9 KONPRT.            3 INSTALLATION SYS-GEN LINES/PAGE
0 MNEMON.            3 PRINT CONTROL: 'AUT'/'MAN'/'NON'
1 MALTH.             3 DEVICE TYPE MNEMONIC FOR ONSITE PRINTERS
2 MSALTM.            3 DAM MAX NO ALT PRT FILES (MALTH <= MSALTM)
3 NCARD,NPAGE,NLINE. 3 SYS-GEN MAX NO ALT PRT FILES (MSALTM = SMALTM)
4 NDHARN,NDFATL,NOTOTL. 3 CARD/PAGE/LINE COUNTERS
5 NDCLRW,NDCLRF.     3 DIAGNOSTIC COUNTERS (SEE NDHARN, NDFATL, ETC.)
6 NWNDOW             3 CLEARANCE COUNTERS
                    3 WINDOW NUMBER:
                    0 = INITIAL VALUE
                    <0 = READY FOR FIRST WINDOW
                    >0 = NEXT WINDOW (OR WINDOW IN PROCESS)

```

END

KONXQT PROC

C-----

```

C*
C*      -- FOR INCLUSION IN EXECUTABLE SUBROUTINES ONLY --
C*      -- MUST NOT BE INCLUDED IN BLOCK DATA SUBROUTINE SYS-BLOCK !!!
C*

```

```

COMMON/KONXQT/
1 MSWTC(2).          3 PROGRAM ID AND STATUS INFO:
2 JMDY,JHMS,JOHM.    3 MODE SWITCHES (SEE DEFINES BELOW)
3 JPRTCN,JRUN(3),JPROG(4),JMDYHM(3).  3 PROGRAM 3XQT DATE/TIME (FIELDATA)
4 JHDO(12.2).        3 PROGRAM HEADING LINE 0
                    3 HEADING LINES 1 & 2

```

DAN PACKAGE APPENDIX Q  
MACROS

XQTLOG-PROCS  
002

```

      5 L INCH,K INCH.      8 DAN LINES/INCH. COLUMNS/INCH
      6 L PAGE,K PAGE.      8 DAN LINES/PAGE. COLUMNS/PAGE
      7 L SINCH.            8 INSTALLATION SYS-GEN LINES/INCH
      8 L SPAGE.            8 INSTALLATION SYS-GEN LINES/PAGE
      9 KONPRT.            8 PRINT CONTROL: 'AUT'/'MAN'/'NON'
      0 MNEON.            8 DEVICE TYPE MNEMONIC FOR ONSITE PRINTERS
      1 HALTH.            8 DAN MAX NO ALT PRT FILES (HALTH <= MSALTH)
      2 MSALTH.            8 SYS-GEN MAX NO ALT PRT FILES (MSALTH = SMALTH)
      3 NCARD,NPAGE,NLINE.  8 CARD/PAGE/LINE COUNTERS
      4 NOWARN,NOFATL,NOTOTL. 8 DIAGNOSTIC COUNTERS (SEE NOWARN, NOFATL, ETC.)
      5 NOCLRW,NOCLRF.      8 CLEARANCE COUNTERS
      6 MWINDOW.           8 WINDOW NUMBER:
                                0 = INITIAL VALUE
                                <0 = READY FOR FIRST WINDOW
                                >0 = NEXT WINDOW (OR WINDOW IN PROCESS)
      7 MWSAV(2)           8 SAVED MODE SWITCHES

C
C
C
C*
      INTEGER JRPIOT(10),JPIOT(7)
      EQUIVALENCE (JRUN(1),JRPIOT(1)), (JPROG(1),JPIOT(1))
C*
      DEFINE MBATCH=FLO(00.6,MSWTCH(1))  8 1=BATCH.      0=DEMAND
      DEFINE MCFIRM=FLO(06.6,MSWTCH(1))  8 1=CONFIRM.  0=DON'T
      DEFINE MCHECK=FLO(12.6,MSWTCH(1))  8 1=CHECKOUT. 0=PRODUCTION
      DEFINE MECHO =FLO(18.6,MSWTCH(1))  8 1=ECHO.      0=DON'T
      DEFINE MDATA=FLO(24.6,MSWTCH(1))  8 1=DATA/CHECKOUT
      DEFINE MADLEV=FLO(30.6,MSWTCH(1))  8 DYNAMIC CSFS 8ADD LEVEL
      DEFINE MLEOND=FLO(00.6,MSWTCH(2))  8 1=LEGEND.   0=NONE
      DEFINE MCOLOR=FLO(06.6,MSWTCH(2))  8 1=COLORCRT. 0=NONE
      DEFINE MLOCAL=FLO(12.6,MSWTCH(2))  8 1=LOCAL.    0=NOT
      DEFINE MPROMT=FLO(18.6,MSWTCH(2))  8 1=PROMPT.   0=DON'T
      DEFINE MDUMP =FLO(24.6,MSWTCH(2))  8 1=DUMP.      0=DON'T
      DEFINE MTRACE=FLO(30.6,MSWTCH(2))  8 1..63=TRACE,0=DON'T
      INTEGER KOMXQT(1)
      EQUIVALENCE (MSWTCH(1),KOMXQT(1))
C*
      END

```

XQTOEF PROC

C-----

```

C*
C*   THIS FUNCTION RETURNS A ZERO OR ONE, INDICATING WHETHER THE FIRST
C*   LETTER OF THE STRING ARGUMENT WAS SPECIFIED AS AN OPTION ON THE 8XQT
C*   CARD. BEFORE REFERENCING THIS FUNCTION, AN ARRAY NAMED JPCT MUST BE
C*   LOADED WITH AT LEAST THE FIRST 16 WORDS OF THE PROGRAM CONTROL TABLE
C*   BY A CALL TO ERPCT.
C*

```

```

      DEFINE XQTOPT(STRING)=FLO((4+STRING/2+30),1,JPCT(16))
C*

```

END

KOMLOG PROC

C-----

C\*

```

COMMON/KOMLOG/ 8 I/O PACKET/BUFFER/POINTERS FOR LOG FILE (UNIT 11)
1 LOPCT.      8 POINTER TO PCT SECTOR
2 LOPLOT.     8 POINTER TO IO/DATE/TIME SECTOR
3 LNOTE.LOHARN.LOFATL. 8 POINTERS TO DIAGNOSTIC SECTOR
4 LTERM.      8 POINTER TO TERMINATION SECTOR
5 LOOPKT(8).  8 I/O PACKET FOR LOG FILE ('I',' ')
6 LBUFR(8).   8 FIRST 8 WORDS OF 25 WORD LOG BUFFER
7 LONSEC.     8 NUMBER OF SECTORS IN LOG FILE (INITIALLY)
8 LOCSF(3).   8 FIRST 3 WORDS OF CSFS STRING (INITIALLY '8','USE 1',' ')
9 LOGFIL(8).  8 EXTERNAL NAME OF LOG FILE FOLLOWED BY ' ' (INITIALLY)
0 LBUFR(9).   8 LAST 9 WORDS OF 25 WORD LOG BUFFER
1 LUGPKT(8)  8 I/O PACKET FOR DIAGNOSTIC MSG QUEUE FILE ('MSGQUEUE')
EQUIVALENCE      8 LOG FILE HEADER
1 (LBUFR(1).LRUNID). 8 EXCLUSIVE ACCESS WORD
2 (LBUFR(2).LOPLO).  8 POINTER TO PCT SECTOR FOR OLDEST PROGRAM
3 (LBUFR(3).LOPHI).  8 POINTER TO PCT SECTOR FOR NEWEST PROGRAM
4 (LBUFR(4).LORHI).  8 POINTER TO PCT SECTOR FOR FIRST PROG OF NEWEST RUN
5 (LBUFR(5).LOMAX)   8 POINTER TO MAXIMUM PCT SECTOR ALLOWED
INTEGER KOMLOG(1)
EQUIVALENCE (LOPCT,KOMLOG(1))

```

C\*

END

PREFACE TO APPENDIX R  
-----

THE FUNCTIONS AND TRANSFORMS IN THIS APPENDIX ARE DESIGNED TO SUPPORT OPERATIONS ON INTEGER STRINGS, CHARACTER STRINGS, CHARACTER BUFFERS, CHARACTERS, BYTE STRINGS, BYTES, AND NYBLES IN AS NEARLY MACHINE-INDEPENDENT A MANNER AS POSSIBLE. ONLY A VERY FEW PRIMITIVE ROUTINES ARE EVEN AWARE OF THE MACHINE-DEPENDENT INTERNAL REPRESENTATION OF THE DATA.

NOMENCLATURE  
-----

INT (INTEGER):

A FORTRAN INTEGER VARIABLE OR CONSTANT CONTAINING THE INTERNAL REPRESENTATION OF A SIGNED FIXED POINT NUMBER WITH A MAGNITUDE LESS THAN  $2^{+31}$ . TO MAINTAIN MACHINE INDEPENDENCE, VALUES OUTSIDE THIS RANGE ORDINARILY SHOULD NOT BE USED.

IST (INTEGER STRING):

A FORTRAN INTEGER ARRAY INTERPRETED AS A STRING OF INTEGERS.

CST (CHARACTER STRING):

A FORTRAN INTEGER ARRAY OR HOLLERITH LITERAL CONTAINING THE PACKED INTERNAL REPRESENTATION OF A STRING OF CHARACTERS IN THE NATIVE CHARACTER CODE OF A MACHINE. THE NUMBER OF BITS PER CHARACTER AND THE NUMBER OF CHARACTERS PER WORD IS KNOWN ONLY TO THOSE PRIMITIVE ROUTINES WHICH EXTRACT OR INSERT INDIVIDUAL CHARACTERS. HOWEVER, THE PACKED CHARACTER MUST NOT CONTAIN MORE THAN 8 SIGNIFICANT BITS (EXCLUDING PAD BITS).

LEN (LENGTH OF CHARACTER STRING OR SUB-STRING)

TWO DIFFERENT LENGTHS ARE ASSOCIATED WITH EACH CHARACTER STRING:

THE PHYSICAL LENGTH OF A CHARACTER STRING IS FIXED AND UNCHANGEABLE -- FOR A CHARACTER STRING CONTAINED IN A (POSSIBLY DIMENSIONED) INTEGER VARIABLE, ITS PHYSICAL LENGTH IS THE MAXIMUM NUMBER OF CHARACTERS WHICH THE VARIABLE CAN HOLD. FOR A CHARACTER STRING DECLARED AS A HOLLERITH LITERAL, ITS PHYSICAL LENGTH IS DETERMINED BY THE COMPILER.

THE LOGICAL LENGTH OF A CHARACTER STRING IS VARIABLE AND DEPENDS UPON THE CURRENT CONTENTS OF THE STRING -- LOGICAL LENGTH IS THE SUM OF ALL LEADING BLANK CHARACTERS, EMBEDDED BLANK CHARACTERS, AND NON-BLANK CHARACTERS WITHIN THE PHYSICAL STRING. LOGICAL LENGTH IGNORES TRAILING BLANK CHARACTERS. (IF THE STRING CONTAINS ONLY BLANK CHARACTERS, THEN THE FIRST BLANK IS CONSIDERED LEADING, AND ALL THE REST TRAILING, YIELDING A LOGICAL LENGTH OF 1 CHARACTER.)

THE LENGTH OF A SUBSTRING WITHIN A STRING IS EXPLICITLY SPECIFIED BY THE PROGRAMMER AND IS INDEPENDENT OF EITHER THE PHYSICAL OR LOGICAL LENGTH OF THE PARENT STRING.

LUC (LOCATION OF INT, CHR, ICE, BYT, NYB, OR SUBSTRING WITHIN STRING)

LOCATIONS IN A STRING ARE ALWAYS NUMBERED IN INCREMENTS OF 1 FROM LEFT TO RIGHT, STARTING WITH 1.



**CB (CHARACTER BUFFER):**

A FORTRAN INTEGER ARRAY CONTAINING THE PACKED INTERNAL REPRESENTATION OF A STRING OF CHARACTERS IN THE NATIVE CHARACTER CODE OF A MACHINE. TOGETHER WITH THE FLAGS AND POINTERS NEEDED FOR BUFFER MANIPULATION. THE ACTUAL REPRESENTATION AND POSITION WITHIN THE BUFFER OF ITS FLAGS AND POINTERS IS KNOWN ONLY TO THE PRIMITIVE BUFFER MANIPULATION ROUTINES.

**CHR (CHARACTER):**

A FORTRAN INTEGER VARIABLE OR CONSTANT CONTAINING THE UNPACKED INTERNAL REPRESENTATION OF A SINGLE CHARACTER IN THE NATIVE CHARACTER CODE OF A MACHINE. LEFT-ALIGNED AND RIGHT-FILLED WITH SPACES. NUMERIC COMPARISONS BETWEEN CHARACTERS SHOULD NOT BE MADE, SINCE THE RESULTS DEPEND ON DIFFERENT MACHINE CONVENTIONS FOR REPRESENTING THE SIGN. THE NATIVE CHARACTER CODE FOR UNIVAC FORTRAN V IS 6-BIT FIELDATA.

**ICE (INTEGER CHARACTER EQUIVALENT):**

A NON-NEGATIVE FORTRAN INTEGER VARIABLE OR CONSTANT CONTAINING THE UNPACKED INTERNAL REPRESENTATION OF A SINGLE CHARACTER IN THE NATIVE CHARACTER CODE OF A MACHINE. SO ALIGNED THAT THE I-C-E CORRESPONDS TO ITS CHARACTER COLLATING SEQUENCE, STARTING FROM ZERO. NUMERIC COMPARISONS MAY BE PERFORMED, BEARING IN MIND THAT THE RESULTS WILL VARY WITH DIFFERENT CHARACTER CODES. FOR UNIVAC FORTRAN V THE I-C-E'S RANGE IS 0 TO 63.

**BST (BYTE STRING):**

A FORTRAN INTEGER ARRAY CONTAINING THE PACKED INTERNAL REPRESENTATION OF A STRING OF BYTES. EACH PACKED BYTE CONTAINS 8 SIGNIFICANT BITS AND MAY ALSO CONTAIN ANY NUMBER OF ZERO-FILLED PAD BITS REQUIRED BY THE HARDWARE TO ACHIEVE EFFICIENT BYTE ACCESS. THE TOTAL NUMBER OF BITS PER PACKED BYTE AND THE NUMBER OF BYTES PER WORD ARE KNOWN ONLY TO THOSE PRIMITIVE ROUTINES WHICH EXTRACT OR INSERT INDIVIDUAL BYTES. ON UNIVAC 1110 SERIES COMPUTERS THERE ARE FOUR 9-BIT PACKED BYTES PER WORD.

**B(N) (EXTERNAL BYTE STRING):**

A FORTRAN INTEGER ARRAY CONTAINING THE PACKED EXTERNAL REPRESENTATION OF A STRING OF BYTES. EACH PACKED BYTE CONTAINS N BITS, OF WHICH NOT MORE THAN 8 ARE SIGNIFICANT. ALL NON-SIGNIFICANT BITS (IF ANY) ARE ZERO-FILLED. MOST COMMONLY, N IS EQUAL TO 8. TO INSURE MACHINE INDEPENDENCE, THE PROGRAMMER SHOULD ASSUME THAT THE EXTERNAL AND INTERNAL REPRESENTATIONS OF BYTE STRINGS ARE DIFFERENT, AND CALL THE APPROPRIATE ROUTINES TO CONVERT THEM AFTER INPUT AND BEFORE OUTPUT.

**BYT (BYTE):**

A NON-NEGATIVE FORTRAN INTEGER VARIABLE OR CONSTANT RESTRICTED TO 8 SIGNIFICANT BITS, SO ALIGNED THAT THE RANGE IS 0 TO +255. TO MAINTAIN MACHINE INDEPENDENCE, VALUES OUTSIDE THIS RANGE MUST NOT BE USED. REGARDLESS OF THE ACTUAL NUMBER OF BITS PER PACKED BYTE ON A PARTICULAR MACHINE, THE CONTENTS OF A BYTE MAY BE INTERPRETED AS THE INTEGER-CHARACTER-EQUIVALENT OF A SINGLE CHARACTER FROM A FOREIGN CHARACTER CODE, AN UNSIGNED INTEGER, OR THE COMPLEMENT REPRESENTATION IN 8 BITS OF A SIGNED INTEGER BETWEEN -127 AND +127.

**NYB (NYBLE):**

A NON-NEGATIVE FORTRAN INTEGER VARIABLE OR CONSTANT RESTRICTED TO 4 SIGNIFICANT BITS, SO ALIGNED THAT THE RANGE IS 0 TO +15. TO MAINTAIN MACHINE INDEPENDENCE, VALUES OUTSIDE THIS RANGE MUST NOT BE USED. REGARDLESS OF THE ACTUAL NUMBER OF BITS PER PACKED NYBLE ON A PARTICULAR MACHINE, THE CONTENTS OF A NYBLE MAY BE INTERPRETED AS A HALF-BYTE, A SMALL UNSIGNED INTEGER, OR THE COMPLEMENT REPRESENTATION IN 4 BITS OF A SIGNED INTEGER BETWEEN -7 AND +7.

**HEX (HEXADECIMAL DIGIT):**

A FORTRAN INTEGER VARIABLE OR CONSTANT CONTAINING THE UNPACKED INTERNAL REPRESENTATION OF A SINGLE CHARACTER WITHIN THE RANGE '0' THRU '9' AND 'A' THRU 'F' IN THE NATIVE CHARACTER CODE OF A MACHINE. LEFT-ALIGNED AND RIGHT-FILLED WITH SPACES. THE CONTENTS OF A HEXADECIMAL DIGIT MAY BE INTERPRETED AS THE CHARACTER REPRESENTATION OF A NYBLE.

**DBY (DOUBLE BYTE):**

A NON-NEGATIVE FORTRAN INTEGER VARIABLE OR CONSTANT RESTRICTED TO 16 SIGNIFICANT BITS, SO ALIGNED THAT THE RANGE IS 0 TO +65,535. TO MAINTAIN MACHINE INDEPENDENCE, VALUES OUTSIDE THIS RANGE MUST NOT BE USED. THE CONTENTS OF A DOUBLE BYTE MAY BE INTERPRETED AS TWO CATENATED BYTES, AN UNSIGNED INTEGER, OR THE COMPLEMENT REPRESENTATION IN 16 BITS OF A SIGNED INTEGER BETWEEN -32,767 AND +32,767.

**QBY (QUADRUPLE BYTE):**

A FORTRAN INTEGER VARIABLE OR CONSTANT RESTRICTED TO 32 SIGNIFICANT BITS. ANY NON-SIGNIFICANT BITS, IF PRESENT, ARE ZERO FILLED. THE SIGN OF A QUADRUPLE BYTE IS MACHINE DEPENDENT AND THEREFORE CONSIDERED UNDEFINED. THE CONTENTS OF A QUADRUPLE BYTE MAY BE INTERPRETED AS FOUR CATENATED BYTES, OR AS THE COMPLEMENT REPRESENTATION IN 32 BITS OF A SIGNED INTEGER WITH A MAGNITUDE LESS THAN  $2^{31}$ .

**KONE (ONE'S COMPLEMENT):**

A FORTRAN INTEGER VARIABLE CONTAINING THE ONE'S COMPLEMENT REPRESENTATION IN A SPECIFIED NUMBER OF SIGNIFICANT BITS (TYPICALLY 4, 8, 16, 32) OF A SIGNED INTEGER. ANY NON-SIGNIFICANT BITS, IF PRESENT, ARE ZERO FILLED. THE SIGN OF ONE'S COMPLEMENT REPRESENTATIONS WITH FEWER THAN 32 BITS IS ALWAYS POSITIVE. THE SIGN OF ONE'S COMPLEMENT REPRESENTATIONS WITH 32 BITS IS MACHINE DEPENDENT AND THEREFORE CONSIDERED UNDEFINED. ONE'S COMPLEMENT REPRESENTATIONS WITH MORE THAN 32 BITS SHOULD NEVER BE USED. ONE'S COMPLEMENT REPRESENTATIONS MUST BE CONVERTED TO ORDINARY INTEGERS WITH NORMAL SIGN EXTENSION BEFORE BEING USED IN ANY ARITHMETIC OPERATIONS.

**KTWO (TWO'S COMPLEMENT):**

A FORTRAN INTEGER VARIABLE CONTAINING THE TWO'S COMPLEMENT REPRESENTATION IN A SPECIFIED NUMBER OF SIGNIFICANT BITS (TYPICALLY 4, 8, 16, 32) OF A SIGNED INTEGER. ANY NON-SIGNIFICANT BITS, IF PRESENT, ARE ZERO FILLED. THE SIGN OF TWO'S COMPLEMENT REPRESENTATIONS WITH FEWER THAN 32 BITS IS ALWAYS POSITIVE. THE SIGN OF TWO'S COMPLEMENT REPRESENTATIONS WITH 32 BITS IS MACHINE DEPENDENT AND THEREFORE CONSIDERED UNDEFINED. TWO'S COMPLEMENT REPRESENTATIONS WITH MORE THAN 32 BITS SHOULD NEVER BE USED. TWO'S COMPLEMENT REPRESENTATIONS MUST BE CONVERTED TO ORDINARY INTEGERS WITH NORMAL SIGN EXTENSION BEFORE BEING USED IN

ANY ARITHMETIC OPERATIONS.  
(TWO'S COMPLEMENT HAS 2 REPRESENTATIONS FOR ZERO: -0 & +0)

USAGE  
-----

TO MINIMIZE MACHINE DEPENDENCE, OPERATIONS INVOLVING A MACHINE'S NATIVE CHARACTER CODE SHOULD ALWAYS UTILIZE THE CST/CHR/ICE FAMILY OF ROUTINES. WHILE THOSE INVOLVING FOREIGN CHARACTER CODES SHOULD ALWAYS UTILIZE THE B<N>/BST/BYT FAMILY OF ROUTINES.

DOUBLE BYTE/CHARACTER FIELDS SHOULD USUALLY START AT AN ODD BYTE/CHARACTER LOCATION.

OTHER MULTIPLE BYTE/CHARACTER FIELDS SHOULD USUALLY START AT BYTE/CHARACTER LOCATIONS 1. 5. 9. ... (4\*N-3) ...

GETINT AND PUTINT ARE PROVIDED FOR COMPATIBILITY IN MANIPULATING STRINGS OF INTEGERS. THEY ARE NOT AS FAST AS CONVENTIONAL SUBSCRIPTING, AND SHOULD ONLY BE USED WHEN NECESSARY.

THE OTHER INT SERIES OF ROUTINES ARE HIGHLY EFFICIENT.

WHEN PASSING A CHARACTER STRING AS AN ARGUMENT TO A FUNCTION OR SUBROUTINE:

- IF THE STRING IS CONTAINED IN A HOLLERITH LITERAL, THE COMPILER (OR PRE-COMPILER) WILL APPEND ONE (NULCHR) OR MORE (NULCST) 'STOP' CHARACTERS TO MARK THE PHYSICAL END OF THE STRING.
- IF THE STRING IS CONTAINED IN AN INTEGER VARIABLE, THE PROGRAMMER MUST EITHER EXPLICITLY PASS THE LENGTH IN A SEPARATE ARGUMENT OR ELSE INSERT A NULCHR OR NULCST TO IMPLICITLY PASS THE LENGTH BY MARKING THE END OF THE STRING.

\*\*\*\*\*  
\* SUBROUTINES PREFIXED WITH AN ASTERISK ARE \*  
\* BEING PHASED OUT AND SHOULD NOT BE USED!! \*  
\*\*\*\*\*

DAM PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

APPENDIX-R  
001

SPRT.SC DAM.PREFACE-R  
SPRT.SC DAM.APPENDIX-R  
SPRT.SC DAM.AS4CST  
SPRT.SC DAM.AS4CB  
SPRT.SC DAM.BST400/1100  
SPRT.SC DAM.BST400/1110  
SPRT.SC DAM.B040ST/1100  
SPRT.SC DAM.B040ST/1110  
SPRT.SC DAM.C0INIT  
SPRT.SC DAM.C04CST  
SPRT.SC DAM.C04FIL  
SPRT.SC DAM.C04IN  
SPRT.SC DAM.C04RL  
SPRT.SC DAM.CLRQWD  
SPRT.SC DAM.CST4AS  
SPRT.SC DAM.CST4CB  
SPRT.SC DAM.CST4IN  
SPRT.SC DAM.CST4RL  
SPRT.SC DAM.CURBST  
SPRT.SC DAM.C0S4CS  
SPRT.SC DAM.C0S4IN  
SPRT.SC DAM.C0S4RL  
SPRT.SC DAM.DCODE  
SPRT.SC DAM.E04AS  
SPRT.SC DAM.E04CST  
SPRT.SC DAM.FIL4CB  
SPRT.SC DAM.GETBYT  
SPRT.SC DAM.GETCHR  
SPRT.SC DAM.GETOBY  
SPRT.SC DAM.GETHEX  
DMSO.N .GETICE  
SPRT.SC DAM.GETINT  
SPRT.SC DAM.GETNUL  
SPRT.SC DAM.GETNYB  
SPRT.SC DAM.GETOBY  
SPRT.SC DAM.GETOKH  
SPRT.SC DAM.GRABSA  
SPRT.SC DAM.OTBYTS  
SPRT.SC DAM.ICE  
SPRT.SC DAM.ICHR  
SPRT.SC DAM.I4KONE  
SPRT.SC DAM.I4KTWO  
SPRT.SC DAM.KMR4IN  
SPRT.SC DAM.KONE4I  
SPRT.SC DAM.KTMO4I  
SPRT.SC DAM.LAPBSA  
SPRT.SC DAM.LBYTEQ  
SPRT.SC DAM.LBYTNE  
SPRT.SC DAM.LCHREQ  
SPRT.SC DAM.LCHRNE  
SPRT.SC DAM.LCSTEQ  
SPRT.SC DAM.LENCST  
SPRT.SC DAM.LENPAD  
SPRT.SC DAM.LICEEQ  
SPRT.SC DAM.LICENE  
SPRT.SC DAM.LINTEQ

. (0000) SET TABS 0 12 & 31  
.  
. ASCII BYTE STRING FOR CHARACTER STRING  
. ASCII BYTE STRING FOR EBCDIC BYTE STRING  
. INTERNAL BYTE STRING FOR 0-BIT EXTERNAL BYTE STR  
. INTERNAL BYTE STRING FOR 0-BIT EXTERNAL BYTE STR  
. 0-BIT EXTERNAL BYTE STRING FOR INTERNAL BYTE STR  
. 0-BIT EXTERNAL BYTE STRING FOR INTERNAL BYTE STR  
. INITIALIZE CHARACTER BUFFER  
. CHARACTER BUFFER FOR CHARACTER STRING  
. CHARACTER BUFFER FOR FILE ('READ')  
. CHARACTER BUFFER FOR INTEGER  
. CHARACTER BUFFER FOR REAL  
. CLEAR QUARTER-WORD MODE  
. CHARACTER STRING FOR ASCII BYTE STRING  
. CHARACTER STRING FOR EBCDIC BYTE STRING  
. CHARACTER STRING FOR INTEGER  
. CHARACTER STRING FOR REAL  
. CURVATURE (1ST DERIVATIVE) OF BYTE STRING  
. VARIABLE-LENGTH (<= 8 CHAR) STRING FOR CHAR STR  
. VARIABLE-LENGTH (<= 8 CHAR) STRING FOR INTEGER  
. VARIABLE-LENGTH (<= 8 CHAR) STRING FOR REAL  
. DECODE NUMERIC CHARACTER STRING  
. EBCDIC BYTE STRING FOR ASCII BYTE STRING  
. EBCDIC BYTE STRING FOR CHARACTER STRING  
. FILE FOR CHARACTER BUFFER ('WRITE')  
. GET NON-NEG INTEGER FROM BYTE IN BYTE STRING  
. GET CHARACTER FROM CHARACTER STRING  
. GET NON-NEG INTEGER FROM DOUBLE BYTE IN BYTE STR  
. GET HEXADECIMAL CHAR FROM NYBLE IN BYTE STRING  
. GET INTEGER-CHAR-EQUIV FROM CHAR STR (SEE GETCHR  
. GET INTEGER FROM INTEGER STRING  
. BEGIN ERROR WALKBACK (ARGS MATCH GETBYT/CHR/INT  
. GET NON-NEG INTEGER FROM NYBLE IN BYTE STRING  
. GET INTEGER FROM QUADRUPLE BYTE IN BYTE STRING  
. GET NEXT CHAR STRING DATA FLD FROM IMAGE BUFFER  
. GRADIENT (1ST DERIVATIVE) OF BYTE STRING ARRAY  
. GET ARRAY OF NON-NEG INTEGERS FROM BYTE STRING  
. INTEGER CHARACTER EQUIVALENT (FROM CHARACTER)  
. CHARACTER (FROM INTEGER CHARACTER EQUIVALENT)  
. INTEGER FOR ONE'S COMPLEMENT  
. INTEGER FOR TWO'S COMPLEMENT  
. \* ENCODE 8-CHAR (FIELDATA) STRING FROM INTEGER  
. ONE'S COMPLEMENT FOR INTEGER  
. TWO'S COMPLEMENT FOR INTEGER  
. LAPLACIAN (2ND DERIVATIVE) OF BYTE STRING ARRAY  
. LOCATION OF BYTE IN STRING EQUAL TO SEARCH BYTE  
. LOCATION OF BYTE IN STRING NOT EQ TO SEARCH BYTE  
. LOCATION OF CHAR IN STRING EQUAL TO SEARCH CHAR  
. LOCATION OF CHAR IN STRING NOT EQ TO SEARCH CHAR  
. LOCATION IN ONE CHARACTER STRING OF ANOTHER  
. LENGTH OF CHARACTER STRING  
. LENGTH PADDED TO NEXT WORD BOUNDARY  
. LOCATION OF ICE IN STRING EQUAL TO SEARCH ICE  
. LOCATION OF ICE IN STRING NOT EQ TO SEARCH ICE  
. LOCATION OF INTEGER IN STRING EQ TO SEARCH INT

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**APPENDIX-R  
002**

SPRT.SC DAN.LINTNE	. LOCATION OF INTEGER IN STRING NE TO SEARCH INT
SPRT.SC DAN.LONCST	. * OF CHAR STRING LOWER IN COLLATING SEQUENCE
SPRT.SC DAN.MOVBST/ASH	. MOVE BYTE STRING
SPRT.SC DAN.MOVBST/FOR	. MOVE BYTE STRING
SPRT.SC DAN.MOVBYT	. MOVE BYTE
SPRT.SC DAN.MOVCHR	. MOVE CHARACTER
SPRT.SC DAN.MOV CST/ASH	. MOVE CHARACTER STRING
SPRT.SC DAN.MOV CST/FOR	. MOVE CHARACTER STRING
SPRT.SC DAN.MOVDBY	. MOVE DOUBLE BYTE
SPRT.SC DAN.MOVIST	. MOVE INTEGER STRING
SPRT.SC DAN.NB4NI	. NUMBER OF BYTES FOR NUMBER OF INTEGERS
SPRT.SC DAN.NC4NI	. NUMBER OF CHARACTERS FOR NUMBER OF INTEGERS
SPRT.SC DAN.NEXTOK	. GET POINTERS TO NEXT TOKEN IN IMAGE BUFFER
SPRT.SC DAN.NI4NB	. NUMBER OF INTEGERS FOR NUMBER OF BYTES
SPRT.SC DAN.NI4NC	. NUMBER OF INTEGERS FOR NUMBER OF CHARACTERS
SPRT.SC DAN.PUTBYT	. PUT NON-NEG INTEGER INTO BYTE OF BYTE STRING
SPRT.SC DAN.PUTCHR	. PUT CHARACTER INTO CHARACTER STRING
SPRT.SC DAN.PUTDBY	. PUT NON-NEG INTEGER INTO DOUBLE BYTE OF BYTE STR
SPRT.SC DAN.PUTHEX	. PUT HEXADECIMAL CHAR INTO NYBLE IN BYTE STRING
SHSO.N .PUTICE	. PUT INTEGER-CHAR-EQUIV INTO CHAR STR (SEE PUTCHR)
SPRT.SC DAN.PUTINT	. PUT INTEGER INTO INTEGER STRING
SPRT.SC DAN.PUTNYB	. PUT NON-NEG INTEGER INTO NYBLE OF BYTE STRING
SPRT.SC DAN.PUTQBY	. PUT INTEGER INTO QUADRUPLE BYTE OF BYTE STRING
SPRT.SC DAN.SETQWD	. SET QUARTER-WORD MODE
SPRT.SC DAN.SLOBST	. SLOPE (1ST DERIVATIVE) OF BYTE STRING
SPRT.SC DAN.TRUEAL	. TRUE IF CST IS ALPHA (26 LTRS + SPACE)
SPRT.SC DAN.TRUCST	. TRUTH VALUE OF CHARACTER STRING COMPARISON

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

AS4CST  
001

```

SUBROUTINE AS4CST( 8 ASCII BYTE STRING FOR CHAR STRING (FIELDATA )
0 JASBUF. 8 ASCII BYTE STRING
.
1 JCSBUF. 8 INTERNAL CHARACTER STRING
1 NCHARS) 8 NUMBER OF CHARACTERS TO CONVERT
      (THE SAME ACTUAL ARGUMENT MAY BE USED FOR JCSBUF & JASBUF)
-----
C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      07/20/78      ORIGINAL CODE
C
C
C METHOD
C -----
C
C      TABLE LOOK-UP.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      TRANSLATE TABLE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      GETICE      8 GET I-C-E FROM CHARACTER STRING
C      PUTBYT      8 PUT BYTE INTO BYTE STRING
C
C
C EXCEPTIONS
C -----
C
C      1. JASBUF IS UNCHANGED IF NBYTES IS LESS THAN 1.
C
C      2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER NCH      8 NUMBER OF CHARACTERS CONVERTED
C      INTEGER NICE      8 I-C-E OF CHARACTER BEING CONVERTED
C      INTEGER JTTA4C(84) 8 TRANSLATE TABLE -- ASCII BYTE FOR CHARACTER
C
C

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

AS4C87  
882

```

C
C      0  1  2  3  4  5  6  7
C      DATA JTTA4C /
C      0  1  2  3  4  5  6  7
C      0  84. 91. 93. 39. 94. 32. 68. 68.
C
C      C  D  E  F  G  H  I  J
C      1  67. 68. 69. 70. 71. 72. 73. 74.
C
C      K  L  M  N  O  P  Q  R
C      2  75. 76. 77. 78. 79. 80. 81. 82.
C
C      S  T  U  V  W  X  Y  Z
C      3  83. 84. 85. 86. 87. 88. 89. 90.
C
C      )  -  *  <  =  >  !  $
C      4  41. 45. 43. 60. 61. 62. 38. 38.
C
C      '  !  &  ;  ?  '  .  \
C      5  42. 40. 37. 58. 63. 33. 44. 92.
C
C      0  1  2  3  4  5  6  7
C      6  48. 49. 50. 51. 52. 53. 54. 55.
C
C      8  9  '  ;  /  .  "
C      7  56. 57. 39. 59. 47. 46. 34. 99/
C
C
C  PROCEDURE
C  -----
C
C
C  CHECK FOR NULL CONVERSION
C
C      IF(INCHARS.LE.0) GO TO 962
C
C
C  TRANSLATE WHILE SCANNING FROM RIGHT TO LEFT
C
C      DC 500 NCH=NCHARS.1.-1
C          CALL GETICE(NICE, JCSBUF.(NCH))
C          CALL PUTBYT(JASBUF.(NCH), JTTA4C(NICE+1))
C  500 CONTINUE
C
C
C  DONE
C
C  900 RETURN
C      END

```

```

SUBROUTINE AS4EB( 8 ASCII BYTE STRING FOR EBCDIC BYTE STRING
0 JASBUF. 8 ASCII BYTE STRING
"
1 JEBBUF. 8 EBCDIC BYTE STRING
1 NBYTES) 8 NUMBER OF BYTES TO CONVERT
      (THE SAME ACTUAL ARGUMENT MAY BE USED FOR JASBUF & JEBBUF)
-----
C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      07/08/78      ORIGINAL CODE
C
C
C METHOD
C -----
C
C      TABLE LOOK-UP.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      GETBYT      8 GET BYTE FROM EBCDIC BYTE STRING
C      PUTBYT      8 PUT BYTE INTO ASCII BYTE STRING
C
C
C EXCEPTIONS
C -----
C
C      1. JASBUF IS UNCHANGED IF NBYTES IS LESS THAN 1.
C
C      2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER NBY      8 NUMBER OF BYTES CONVERTED
C      INTEGER IBYT     8 BYTE BEING CONVERTED
C      INTEGER JTABLE(256) 8 TRANSLATE TABLE -- ASCII FOR EBCDIC
C
C

```



DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

AS4CB  
002

```

C
C      0 1 2 3 4 5 6 7 8 9 A B C D E F
C      DATA (JTTA4E(N),N=1,64) /      8 CONTROL CHARACTERS
C      NUL SOH STX ETX HT DEL VT FF CR SO SI
C      0 000.001.002.003. 0.009. 0.127. 0. 0. 0.011.012.013.014.015.
C
C      DLE DC1 DC2 DC3 BS CAN EM FS GS RS US
C      1 016.017.018.019. 0. 0.008. 0.024.025. 0. 0.028.029.030.031.
C
C      LF ETB ESC ENQ ACK BEL
C      2 0. 0. 0. 0. 0.010.023.027. 0. 0. 0. 0. 0.005.006.007.
C
C      SYN RS EOT DC4 NAK SUB
C      3 0. 0.022. 0. 0.030. 0.004. 0. 0. 0. 0.020.021. 0.026/
C
C      0 1 2 3 4 5 6 7 8 9 A B C D E F
C      DATA (JTTA4E(N),N=65,128) /      8 PUNCTUATION
C      SP . < ( + OR
C      4 032. 32. 32. 32. 32. 32. 32. 32. 32. 32. 32.046.060.040.043.124.
C
C      & ! $ % ' : NOT
C      5 038. 32. 32. 32. 32. 32. 32. 32. 32. 32. 32.033.036.042.041.059.094.
C
C      - / . : --- > ?
C      6 045.047. 32. 32. 32. 32. 32. 32. 32. 32. 32.044.037.095.062.063.
C
C      ACCENT : ° ¢ ' ° ¢
C      7 32. 32. 32. 32. 32. 32. 32. 32. 32.096.058.035.064.039.061.034/
C
C      0 1 2 3 4 5 6 7 8 9 A B C D E F
C      DATA (JTTA4E(N),N=129,192) /      8 LOWER CASE
C      A B C D E F G H I
C      8 32.097.098.099.100.101.102.103.104.105. 32. 32. 32. 32. 32. 32.
C
C      J K L M N O P Q R
C      9 32.106.107.108.109.110.111.112.113.114. 32. 32. 32. 32. 32. 32.
C
C      TILD S T U V W X Y Z I
C      A 32.126.115.116.117.118.119.120.121.122. 32. 32. 32.091. 32. 32.
C
C      I
C      B 32. 32. 32. 32. 32. 32. 32. 32. 32. 32. 32. 32. 32.093. 32. 32/
C
C      0 1 2 3 4 5 6 7 8 9 A B C D E F
C      DATA (JTTA4E(N),N=193,256) /      8 UPPER CASE
C      -I A B C D E F G H I
C      C 123.065.066.067.068.069.070.071.072.073. 32. 32. 32. 32. 32. 32.
C
C      I- J K L M N O P Q R
C      D 125.074.075.076.077.078.079.080.081.082. 32. 32. 32. 32. 32. 32.
C
C      \ S T U V W X Y Z
C      E 092. 32.083.084.085.086.087.088.089.090. 32. 32. 32. 32. 32. 32.
C
C      0 1 2 3 4 5 6 7 8 9
C      F 048.049.050.051.052.053.054.055.056.057. 32. 32. 32. 32. 32. 32/

```

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**AS4C0  
003**

```
C
C
C PROCEDURE
C -----
C
C
C CHECK FOR NULL CONVERSION
C
C     IF(NBYTES.LE.0) GO TO 900
C
C
C TRANSLATE WHILE SCANNING FROM LEFT TO RIGHT
C
C     DO 500 NBY=1,NBYTES
C         CALL GETBYT(1BYT,  JEBBUF,(NBY))
C         CALL PUTBYT(JASBUF,(NBY),  JTTA4E(1BYT+1))
C     500 CONTINUE
C
C
C DONE
C
C     900 RETURN
C         END
```

. SUBROUTINE BST400( 8 INTERNAL BYTE STRING FOR 8-BIT BYTE STRING (1100)  
. 0 IOST. 8 INTERNAL BYTE STRING  
. .  
. 1 IBS8. 8 EXTERNAL BYTE STRING OF 8-BIT BYTES  
. 1 NBYTES) 8 NUMBER OF BYTES TO TRANSFORM  
. (THE SAME ACTUAL ARGUMENT MAY BE USED FOR IOST & IBS8)  
. -----  
. .

. HISTORY  
. -----  
. .

. E H SCHLOSSER LEC 07/06/78 REQUIREMENTS  
. E H SCHLOSSER LEC 08/01/79 DESIGN/CODE/TEST  
. .

. METHOD  
. -----  
. .

. COMPUTE NUMBER OF QUADRUPLE BYTES TO TRANSFORM, INCLUDING PARTIAL  
. QUADRUPLE BYTES. COMPUTE NUMBER OF BLOCKS (9 QUADBYTES EACH) TO  
. TRANSFORM, INCLUDING PARTIAL BLOCKS. COMPUTE NUMBER OF QUADRUPLE  
. BYTES TO TRANSFORM IN RIGHTMOST BLOCK. DO TRANSFORMATION FROM RIGHT  
. TO LEFT, STARTING WITH PARTIAL BLOCK (IF ANY).  
. .

. MACHINE-DEPENDENT CODE  
. -----  
. .

. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 COMPUTER. THE METHOD  
. OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V. IMPLEMENTING  
. CODE MUST BE REWRITTEN FOR DIFFERENT COMPILERS (E.G., UNIVAC ASCII  
. FORTRAN) AND DIFFERENT MACHINES.  
. .

. EXTERNAL REFERENCES  
. -----  
. .

. NONE.  
. .

. EXCEPTIONS  
. -----  
. .

- . 1. IOST IS UNCHANGED IF NBYTES IS LESS THAN 1.
- . 2. THE TRANSFORMATION INTO INTERNAL BYTE STRING FROM EXTERNAL 8-BIT  
. BYTE STRING IS PERFORMED IN GROUPS OF 4 BYTES (1 QUADRUPLE-BYTE) AT A  
. TIME. THUS, IF NBYTES IS NOT EVENLY DIVISIBLE BY 4, THE ACTUAL NUMBER  
. OF BYTES TRANSFORMED WILL BE THE NEXT HIGHER NUMBER WHICH IS EVENLY  
. DIVISIBLE BY 4.
- . 3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

. GLOBAL DECLARATIONS  
. .

```

.-----
.
.          AXRS          . STANDARD UNIVAC 1100 REGISTER MNEMONICS
.
.
. LOCAL DECLARATIONS
.-----
.
BQAS*      PROC          . PROC TO SIMULATE 1110 BQ AS INSTR ON 1100
              LA          A4.A5      .
              DSL          A4.8       .
              SSL          A5.1       .
              DSL          A4.8       .
              SSL          A5.1       .
              DSL          A4.8       .
              SSL          A5.1       .
              DSL          A4.8       .
              SSL          A5.1       .
              END          .

.
. PROCEDURE
.-----
.
S(00) . 1-BANK          . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
BST488*      LA          A2.*2.X11   . # OF BYTES TO TRANSFORM
              LA.U        A1.0        . DIVISION COMING
              AA.U        A2.3        .
              DI.U        A1.4        . A1 := # OF QUADBYTES TO TRANSFORM
              .           . A2 := (#-1) OF BYTES TO TRANSF IN RIGHTMOST QUADBYTE

.
              LA.U        A0.0        . DIVISION COMING
              AA.U        A1.8        .
              DI.U        A0.9        . A0 := # OF BLOCKS TO TRANSFORM
              .           . A1 := (#-1) OF QUADBYTES TO TRANSF IN RIGHTMOST BLOCK
              ANA.U       A0.1        . A0 := (#-1) OF BLOCKS TO TRANSFORM
              JN          A0.RETURN   . RETURN IF NO BLOCKS

.
              LA          A2.A0      .
              LA          A3.A0      .
              MSI.U       A2.8        . 8 WORDS PER INPUT BLOCK
              MSI.U       A3.9        . 9 WORDS PER OUTPUT BLOCK
              AA          A2.1.X11   . A2 := ADDR OF RIGHTMOST INPUT BLOCK
              AA          A3.0.X11   . A3 := ADDR OF RIGHTMOST OUTPUT BLOCK
              LXI.XU      A2.-8      . INPUT BLOCK ADDR INCREMENT IS -8
              LXI.XU      A3.-9      . OUTPUT BLOCK ADDR INCREMENT IS -9
              J           TRANSF.A1  . START TRANSFORMING AT PROPER QUADBYTE!!!

.
TRANSF      J           TRQBY1      .
              J           TRQBY2      .
              J           TRQBY3      .
              J           TRQBY4      .
              J           TRQBY5      .

```

DAM PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

BS7486/1100  
003

	J	TRQBY6	.
	J	TRQBY7	.
	J	TRQBY8	.
TRQBY9	LA	A5.7.A2	.
	BQAS		.
	SA	A5.8.A3	.
TRQBY8	OL	A4.6.A2	.
	DSL	A4.32	.
	BQAS		.
	SA	A5.7.A3	.
TRQBY7	OL	A4.5.A2	.
	DSL	A4.28	.
	BQAS		.
	SA	A5.6.A3	.
TRQBY6	OL	A4.4.A2	.
	DSL	A4.24	.
	BQAS		.
	SA	A5.5.A3	.
TRQBY5	OL	A4.3.A2	.
	DSL	A4.20	.
	BQAS		.
	SA	A5.4.A3	.
TRQBY4	OL	A4.2.A2	.
	DSL	A4.16	.
	BQAS		.
	SA	A5.3.A3	.
TRQBY3	OL	A4.1.A2	.
	DSL	A4.12	.
	BQAS		.
	SA	A5.2.A3	.
TRQBY2	OL	A4.0.A2	.
	DSL	A4.8	.
	BQAS		.
	SA	A5.1.A3	.
TRQBY1	LA	A5.0.*A2	.
	SSL	A5.4	.
	BQAS		.
	SA	A5.0.*A3	.
	JOD	A0.TRQBY9	.
.			
RETURN	J	4.X11	.
	END		

SUBROUTINE BS7400( I INTERNAL BYTE STRING FOR 8-BIT BYTE STRING (1110)  
O 1BST, I INTERNAL BYTE STRING  
I 1BSB, I EXTERNAL BYTE STRING OF 8-BIT BYTES  
I NBYTES) I NUMBER OF BYTES TO TRANSFORM  
(THE SAME ACTUAL ARGUMENT MAY BE USED FOR 1BST & 1BSB)  
-----

HISTORY  
-----

E H SCHLOSSER	LEC	07/06/78	REQUIREMENTS
E H SCHLOSSER	LEC	08/01/79	DESIGN/CODE/TEST

METHOD  
-----

COMPUTE NUMBER OF QUADRUPLE BYTES TO TRANSFORM, INCLUDING PARTIAL  
QUADRUPLE BYTES. COMPUTE NUMBER OF BLOCKS (8 QUADBYTES EACH) TO  
TRANSFORM, INCLUDING PARTIAL BLOCKS. COMPUTE NUMBER OF QUADRUPLE  
BYTES TO TRANSFORM IN RIGHTMOST BLOCK. DO TRANSFORMATION FROM RIGHT  
TO LEFT, STARTING WITH PARTIAL BLOCK (IF ANY).

MACHINE-DEPENDENT CODE  
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1110 COMPUTER USING SPECIAL BYTE-  
MANIPULATION HARDWARE NOT PRESENT IN THE UNIVAC 1108. THE METHOD  
OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V. IMPLEMENTING  
CODE MUST BE REWRITTEN FOR DIFFERENT COMPILERS (E.G., UNIVAC ASCII  
FORTRAN) AND DIFFERENT MACHINES.

EXTERNAL REFERENCES  
-----

NONE.

EXCEPTIONS  
-----

1. 1BST IS UNCHANGED IF NBYTES IS LESS THAN 1.
2. THE TRANSFORMATION INTO INTERNAL BYTE STRING FROM EXTERNAL 8-BIT  
BYTE STRING IS PERFORMED IN GROUPS OF 4 BYTES (1 QUADRUPLE-BYTE) AT A  
TIME. THUS, IF NBYTES IS NOT EVENLY DIVISIBLE BY 4, THE ACTUAL NUMBER  
OF BYTES TRANSFORMED WILL BE THE NEXT HIGHER NUMBER WHICH IS EVENLY  
DIVISIBLE BY 4.
3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

DAM PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

BST488/1110  
002

. GLOBAL DECLARATIONS  
. -----

AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS  
. -----

BQAS\* PROC . 1110 BQ AS INSTRUCTION CODED FOR 1100 ASSEMBLER  
INSTRF FORM 6.4.4.4.2.16 . INSTRUCTION FORM  
INSTRF 037.01.05.0.0.0 . BQ AS  
END .

. PROCEDURE  
. -----

\$1001 . 1-BANK . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY  
BST488\* LA A2.42.X11 . # OF BYTES TO TRANSFORM  
LA.U A1.0 . DIVISION COMING  
AA.U A2.3 .  
DI.U A1.4 . A1 := # OF QUADBYTES TO TRANSFORM  
A2 := (#-1) OF BYTES TO TRANSF IN RIGHTMOST QUADBYTE

LA.U A0.0 . DIVISION COMING  
AA.U A1.0 .  
DI.U A0.9 . A0 := # OF BLOCKS TO TRANSFORM  
A1 := (#-1) OF QUADBYTES TO TRANSF IN RIGHTMOST BLOCK  
ANA.U A0.1 . A0 := (#-1) OF BLOCKS TO TRANSFORM  
JM A0.RETURN . RETURN IF NO BLOCKS

LA A2.A0 .  
LA A3.A0 .  
MS1.U A2.0 . 8 WORDS PER INPUT BLOCK  
MS1.U A3.9 . 9 WORDS PER OUTPUT BLOCK  
AA A2.1.X11 . A2 := ADDR OF RIGHTMOST INPUT BLOCK  
AA A3.0.X11 . A3 := ADDR OF RIGHTMOST OUTPUT BLOCK  
LX1.XU A2.-0 . INPUT BLOCK ADDR INCREMENT IS -8  
LX1.XU A3.-9 . OUTPUT BLOCK ADDR INCREMENT IS -9  
J TRANSF.A1 . START TRANSFORMING AT PROPER QUADBYTE!!!

TRANSF J TRQBY1 .  
J TRQBY2 .  
J TRQBY3 .  
J TRQBY4 .  
J TRQBY5 .  
J TRQBY6 .  
J TRQBY7 .  
J TRQBY8 .  
TRQBY9 LA A5.7.A2 .  
BQAS SA A5.8.A3 .  
SA

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

OST400/1110  
003

TRQBY6	DL	A4.6.A2	.
	DSL	A4.32	.
	BQAS		.
	SA	A5.7.A3	.
TRQBY7	DL	A4.9.A2	.
	DSL	A4.28	.
	BQAS		.
	SA	A5.8.A3	.
TRQBY8	DL	A4.4.A2	.
	DSL	A4.24	.
	BQAS		.
	SA	A5.9.A3	.
TRQBY9	DL	A4.3.A2	.
	DSL	A4.20	.
	BQAS		.
	SA	A5.4.A3	.
TRQBY4	DL	A4.2.A2	.
	DSL	A4.16	.
	BQAS		.
	SA	A5.3.A3	.
TRQBY3	DL	A4.1.A2	.
	DSL	A4.12	.
	BQAS		.
	SA	A5.2.A3	.
TRQBY2	DL	A4.0.A2	.
	DSL	A4.8	.
	BQAS		.
	SA	A5.1.A3	.
TRQBY1	LA	A5.0.*A2	.
	SSL	A5.4	.
	BQAS		.
	SA	A5.0.*A3	.
	JGD	A0.TRQBY9	.
.			
RETURN	J	4.X11	.
	END		



DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

BB4BST/1100  
001

SUBROUTINE BB4BST( 8 8-BIT BYTE STRING FOR INTERNAL BYTE STRING (1100)  
0 1000. 8 EXTERNAL BYTE STRING OF 8-BIT BYTES  
1 1001. 8 INTERNAL BYTE STRING  
1 NBYTES) 8 NUMBER OF BYTES TO CONVERT  
(THE SAME ACTUAL ARGUMENT MAY BE USED FOR 1001 & 1000)  
-----

HISTORY  
-----

E H SCHLOSSER LEC 07/06/78 ORIGINAL REQUIREMENTS

METHOD  
-----

COMPRESS ADJACENT 9-BIT BYTES INTO 8-BITS EACH. AND THEN  
SHIFT INTO PROPER ALIGNMENT.

MACHINE-DEPENDENT CODE  
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 COMPUTER. THE METHOD  
OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V. IMPLEMENTING  
CODE MUST BE REWRITTEN FOR DIFFERENT COMPILERS (E.G., UNIVAC ASCII  
FORTRAN) AND DIFFERENT MACHINES.

EXTERNAL REFERENCES  
-----

NONE

EXCEPTIONS  
-----

1. 1000 IS UNCHANGED IF NBYTES IS LESS THAN 1.
2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

GLOBAL DECLARATIONS  
-----

NONE.

AXRS . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES  
S(00) . 1-BANK  
BB4BST.  
\*\*\*\*\* NOT YET IMPLEMENTED \*\*\*\*\*

```

SUBROUTINE BB4BST( 8 8-BIT BYTE STRING FOR INTERNAL BYTE STRING (1110)
0 1BSB) 8 EXTERNAL BYTE STRING OF 8-BIT BYTES
.
.
1 1BST. 8 INTERNAL BYTE STRING
1 NBYTES) 8 NUMBER OF BYTES TO CONVERT
.
.
. (THE SAME ACTUAL ARGUMENT MAY BE USED FOR 1BST & 1BSB)
.
. -----

```

# HISTORY

-----

E M SCHLOSSER LEC 07/06/78 ORIGINAL CODE

# METHOD

-----

COMPRESS ADJACENT 9-BIT BYTES INTO 8-BITS EACH, AND THEN  
SHIFT INTO PROPER ALIGNMENT.

# MACHINE-DEPENDENT CODE

-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1110 COMPUTER USING SPECIAL BYTE-  
MANIPULATION HARDWARE NOT PRESENT IN THE UNIVAC 1100. THE METHOD  
OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V. IMPLEMENTING  
CODE MUST BE REWRITTEN FOR DIFFERENT COMPILERS (E.G., UNIVAC ASCII  
FORTRAN) AND DIFFERENT MACHINES.

# EXTERNAL REFERENCES

-----

NONE

# EXCEPTIONS

-----

1. 1BSB IS UNCHANGED IF NBYTES IS LESS THAN 1.
2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
3. SINCE THE ACTUAL CONVERSION IS PERFORMED IN GROUPS OF 8 BYTES AT  
A TIME, UP TO 7 MORE BYTES THAN REQUESTED MAY BE CONVERTED. 1BST  
AND 1BSB SHOULD BE LARGE ENOUGH TO ACCOMMODATE THIS.

# GLOBAL DECLARATIONS

-----

AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

0040ST/1110  
002

. LOCAL DECLARATIONS

. -----

S(00) . 0-BANK

X14XMS

XSAVE

RES

4.8

3

. X INCREMENT = 4. X MODIFIER = 0

. SAVE AREA FOR X REOS 2.3.4

. PROCEDURE

. -----

S(01) . 1-BANK

0040ST

SX

X2.XSAVE+0

. SAVE REOS

SX

X3.XSAVE+1

.

SX

X4.XSAVE+2

.

LA

A2.0.X11

. NUMBER OF BYTES TO CONVERT

LX

X2.1.X11

. ADDRESS OF 8-BIT INPUT BUFFER

AA.XU

A2.-1

. NUMBER OF BYTES - 1

LX

X4.0.X11

. ADDRESS OF 8-BIT OUTPUT BUFFER

IN

A2.RETURN

. NO BYTES TO CONVERT

SSL

A2.2

. NUMBER OF INPUT WORDS - 1

LX1.XU

X2.1

. AUTO INCREMENT FOR INPUT IS 1

LX1.XU

X4.1

. AUTO INCREMENT FOR OUTPUT IS 1

J

FIRSTCYCLE

.

NEXTCYCLE

J00

A3.NEXTHORD

. DECREMENT SHIFTS & RECYCLE IF ZERO

AX.XU

X2.1

. CONTENTS AT ADDR X2 ALREADY EXHAUSTED

AA.XU

A2.-1

.

FIRSTCYCLE

LX

X3.X14XMS

. NUMBER OF BITS TO SHIFT IS 0.12. ...

LA.XU

A3.7

. EIGHT SHIFTS PER CYCLE

NEXTHORD

DL

A0.0.0.X2

. LOAD 2 WDS INPUT & INCREMENT ADDR BY 1

IFORM

Q00

A0

. COMPRESS 8 BYTES IN ADDR1 (HAND-CODED)

FORM 0.4.4.4.2.16 . INSTRUCT FORM

IFORM 037.04.0.0.0.0 . Q00 A0 INSTRUCT

LOSL

A0.0.0.X3

. SHIFT 4.8 BYTES INTO A0

SA

A0.0.0.X4

. STORE OUTPUT WD & INCREMENT ADDR

J00

A2.NEXTCYCLE

. DECREMENT INPUT WDS LEFT & REPEAT

LX

X2.XSAVE+0

. RESTORE REOS

LX

X3.XSAVE+1

.

LX

X4.XSAVE+2

.

RETURN

J

4.X11

.

END

```

SUBROUTINE CBINIT( 8 INITIALIZE CHARACTER BUFFER
8 KBOUT) 8 OUTPUT CHARACTER BUFFER
-----
C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      09/18/79      REQUIREMENTS
C      E M SCHLOSSER      LEC      09/27/79      DESIGN & CODE
C
C
C METHOD
C -----
C
C      INITIALIZE BUFFER NULCST & POINTERS.
C      MOVE NULCHR. PADDED WITH BLANKS. TO STRING PORTION OF BUFFER.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      MOVCSY      8 MOVE CHARACTER STRING
C      INTEGER NC4NI 8 NUMBER OF CHARACTERS FOR NUMBER OF INTEGERS
C
C
C EXCEPTIONS
C -----
C
C      1. IF KBOUT IS SMALLER IN SIZE THAN A STANDARD CHARACTER BUFFER
C      (SUCH AS ICBUF1, ICBUF2, ETC.), THEN THE RESULTS ARE UNDEFINED.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      INCLUDE CDEF.LIST      8 DEFINE STRUCTURE OF CHARACTER BUFFERS
C      INCLUDE NULCST.LIST    8 DEFINE NULL CHARACTER STRING
C      INCLUDE NULCHR.LIST    8 DEFINE NULL CHARACTER
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER KBOUT(CBSZIN) 8 ARGUMENT
C
C
C PROCEDURE
C -----
C

```

**DAH PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**COINIT  
002**

```
C
C INITIALIZE NULCST, LOCATION POINTER, END POINTER
C
    KBOUT(CBNUL)=NULCST
    KBOUT(CBLOC)=0
    KBOUT(CBEND)=NC4N1(CBNUL-1)
C
C
C INITIALIZE NON-POINTER PORTION OF BUFFER TO 1 NULCHR PADDED WITH SPACES
C
    CALL MOVCST(KBOUT,11,(KBOUT(CBEND)),
    *          NULCHR,11,11,' ')
C
C
C DONE
C
    RETURN
    END
```

```

SUBROUTINE CB4CST( 3 APPEND CHARACTERS TO CHAR BUFFER FROM CHAR STRING
0 KBOUT, 3 OUTPUT CHARACTER BUFFER
-
1 KSTIN, 3 INPUT CHARACTER STRING
( LOCIN, 3 (OPTIONAL) CHAR LOC WITHIN INPUT STRING WHERE SUBSTRING BEGINS
( LENIND, 3 (OPTIONAL) LENGTH & DIRECTION OF INPUT SUBSTRING
3 3.3) 3 TO KEEP COMPILER STRAIGHT -- DON'T USE!!!
(CHAR LOCATION COUNTED FROM 1 AT LEFT OF STRING)
(POSITIVE DIRECTION = LEFT TO RIGHT)
-----
C
C
C
C
C HISTORY
C -----
C
C E H SCHLOSSER LEC 09/12/79 REQUIREMENTS
C E H SCHLOSSER LEC 09/27/79 DESIGN & CODE
C
C
C METHOD
C -----
C
C DETERMINE ACTUAL/IMPLIED VALUES FOR OPTIONAL ARGUMENTS.
C MOVE INPUT SUBSTRING TO FIRST AVAILABLE LOCATION IN OUTPUT CHARACTER
C BUFFER AND UPDATE BUFFER POINTER.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C UNIVAC FORTRAN V RETURN K.
C
C
C EXTERNAL REFERENCES
C -----
C
C ARORET 3 DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR
C MOVCST 3 MOVE CHARACTER STRING
C INTEGER LENCST 3 LENGTH OF CHARACTER STRING
C INTEGER IOUP 3 INTEGER DUPLICATE (INDIRECT REFERENCE TO OPTIONAL ARG)
C
C
C EXCEPTIONS
C -----
C
C 1. IF KBOUT IS SMALLER IN SIZE THAN A STANDARD CHARACTER BUFFER
C SUCH AS ICBUF1, ICBUF2, ETC.) THEN THE RESULTS ARE UNDEFINED.
C
C 2. IF KBOUT HAS NOT BEEN PREVIOUSLY INITIALIZED BY CBINIT THE RESULTS
C ARE UNDEFINED.
C
C 3. IF THE INPUT SUBSTRING LIES WHOLLY OR PARTIALLY OUTSIDE ITS STRING THE
C RESULTS WILL BE UNDEFINED.
C
C 4. IF THE OPTIONAL ARGUMENTS LOCIN & LENIND ARE OMITTED, THEN KSTIN
C MUST BE A VARIABLE-LENGTH CHARACTER STRING (A HOLLERITH LITERAL.

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CB4CST  
002

```

C      OR AN ARRAY OF PACKED CHARACTERS TERMINATED WITH NULCHR OR NULCST).
C
C      5. IF THERE IS INSUFFICIENT ROOM REMAINING IN THE OUTPUT BUFFER, THEN
C      ONLY PART OF THE INPUT STRING WILL BE APPENDED TO THE BUFFER AND
C      THE REMAINDER IGNORED.
C
C      GLOBAL DECLARATIONS
C      -----
C
C      INCLUDE CDEF.LIST      % DEFINE STRUCTURE OF CHARACTER BUFFERS
C      INCLUDE NULCST.LIST    % DEFINE NULL CHARACTER STRING
C
C      LOCAL DECLARATIONS
C      -----
C
C      INTEGER KBOUT(CBSZIN)      % ARGUMENT
C      INTEGER NARGO              % NUMBER OF ACTUAL ARGUMENTS
C      INTEGER KRETN              % RETURN K VECTOR
C      INTEGER LENOUT             % LENGTH (NUMBER OF CHARACTERS) TO APPEND
C      INTEGER LOCINN             % CHAR LOC WITHIN INPUT STRING WHERE SUBSTRING BEGINS
C      INTEGER LENINN             % LENGTH & DIRECTION OF INPUT SUBSTRING
C
C
C      C PROCEDURE
C      -----
C
C      DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR
C
C      % IF(KSTIN.EQ.'') CALL XREQ77(2)  %-% *** DEBUG ***  %-%
C      CALL ARQRET(NARGO,%KRETN)
C
C      DETERMINE VALUES FOR OPTIONAL ARGUMENTS
C
C      LOCINN=1
C      IF(NARGO.GT.2) LOCINN=IDUP(LOCIN)
C      IF(NARGO.LE.3) LENINN=LENCST(KSTIN,132)-LOCINN+1
C      IF(NARGO.GT.3) LENINN=IDUP(LENIND)
C
C
C      C CHECK BUFFER POINTERS
C
C      IF(KBOUT(CBNUL).NE.NULCST) GO TO 970
C      IF(KBOUT(CBLOC).LT.0) GO TO 900
C      IF(KBOUT(CBEND).GT.255) GO TO 900
C
C
C      C COMPUTE LENGTH TO APPEND TO BUFFER (IF ANY)
C
C      LENOUT=MIN0(KBOUT(CBEND)-KBOUT(CBLOC),ABS(LENINN))
C      IF(LENOUT.LE.0) GO TO 900
C
C

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CS4CST  
003

```
C MOVE SUBSTRING TO FIRST AVAILABLE LOCATION IN BUFFER
C
    CALL MOVCST(KBOUT.(KBOUT(CBLOC)+1).(LENOUT).
-          KSTIN.(LOCINN).(LENINN). ' ')
C
C
C UPDATE BUFFER LOC TO POINT TO LAST CHARACTER IN BUFFER
C
    KBOUT(CBLOC)=KBOUT(CBLOC)+LENOUT
/
C
CS900 RETURN KRETN
900 CONTINUE 3-3 DEBUG 3-3
C S IF(KSTIN.EQ.' ') CALL XREG77(2) 3-3 *** DEBUG *** 3-3
    RETURN KRETN 3-3 DEBUG 3-3
END
```



DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CONFIL  
001

(NOT IMPLEMENTED)

```

SUBROUTINE CB4IN( 8 ENCODE INTEGER & APPEND TO CHARACTER BUFFER
O KBOUT. 8 OUTPUT CHARACTER BUFFER
-
I INTOER. 8 INTEGER TO ENCODE
I MINFLW. 8 (OPTIONAL) MINIMUM FIELD WIDTH (12 IF OMITTED)
I LDZCHR. 8 (OPTIONAL) LEADING ZERO CHARACTER (BLANK IF OMITTED)
S 8.S) 8 TO KEEP COMPILER STRAIGHT -- DON'T USE!!!
-----
C
C
C
C HISTORY
C -----
C
C E H SCHLOSSER LEC 09/28/79 REQUIREMENTS
C E H SCHLOSSER LEC 10/05/79 DESIGN & CODE
C
C
C METHOD
C -----
C
C DETERMINE ACTUAL/IMPLIED VALUES FOR OPTIONAL ARGUMENTS.
C ENCODE INTEGER WITH MINIMUM FIELDWIDTH OF MINFLW & MAXIMUM FIELDWIDTH
C OF 12. MOVE ENCODED STRING TO FIRST AVAILABLE LOCATION IN OUTPUT
C CHARACTER BUFFER AND UPDATE BUFFER POINTER. THIS TRANSFORM PROVIDES
C EXTENDED PASCAL-TYPE INTEGER ENCODING.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C UNIVAC FORTRAN V RETURN K.
C
C
C EXTERNAL REFERENCES
C -----
C
C ARGRET 8 DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR
C CST4IN 8 CHARACTER STRING FOR INTEGER
C INTEGER IDUP 8 INTEGER DUPLICATE (INDIRECT REFERENCE TO OPTIONAL ARG)
C
C
C EXCEPTIONS
C -----
C
C 1. IF KBOUT IS SMALLER IN SIZE THAN A STANDARD CHARACTER BUFFER
C (SUCH AS ICBUF1, ICBUF2, ETC.) THEN THE RESULTS ARE UNDEFINED.
C
C 2. IF KBOUT HAS NOT BEEN PREVIOUSLY INITIALIZED BY CBINIT THE RESULTS
C ARE UNDEFINED.
C
C 3. IF THERE IS INSUFFICIENT ROOM REMAINING IN THE OUTPUT BUFFER, THEN
C ONE OR MORE '?' WILL BE APPENDED TO THE BUFFER, IF POSSIBLE.
C
C
C GLOBAL DECLARATIONS
C -----

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CB41N  
002

```

C
C      INCLUDE CDEF.LIST      & DEFINE STRUCTURE OF CHARACTER BUFFER
C      INCLUDE NULCST.LIST   & DEFINE NULL CHARACTER STRING
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER KBOUT(1)      & ARGUMENT
C      INTEGER NAROS         & NUMBER OF ACTUAL ARGUMENTS
C      INTEGER KRETN         & RETURN K VECTOR
C      INTEGER MINIFW        & MINIMUM FIELD WIDTH
C      INTEGER LEADZC        & LEADING ZERO CHARACTER
C
C
C PROCEDURE
C -----
C
C DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR
C
C      CALL ARGRET(NAROS,KRETN)
C
C DETERMINE VALUES FOR OPTIONAL ARGUMENTS
C
C      MINIFW=12
C      IF(NAROS.GE.3) MINIFW=IDUP(MINIFW)
C      LEADZC=' '
C      IF(NAROS.GE.4) LEADZC=IDUP(LDZCHR)
C
C
C CHECK BUFFER POINTERS
C
C      IF(KBOUT(CBNUL).NE.NULCST) GO TO 900
C      IF(KBOUT(CBLOC).LT.0) GO TO 900
C      IF(KBOUT(CBEND).GT.255) GO TO 900
C
C
C COMPUTE MAXIMUM LENGTH TO APPEND TO BUFFER. IF ANY
C
C      LENMAX=MIN0(KBOUT(CBEND)-KBOUT(CBLOC),MAX0(MINIFW,12))
C      IF(LENMAX.LE.0) GO TO 900
C
C
C ENCODE INTEGER AT FIRST AVAILABLE LOCATION IN BUFFER
C
C      CALL CST4IN(KBOUT,(KBOUT(CBLOC)+1),(LENMAX),
C      .          ,INTOER,MINIFW,LEADZC)
C
C
C UPDATE BUFFER LOC TO POINT TO LAST CHARACTER NOW IN BUFFER
C
C      KBOUT(CBLOC)=LENCST(KBOUT,(KBOUT(CBLOC)+LENMAX))
C
C

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CD41N  
003

C DONE

C

900 RETURN KRETN  
END

```

C -----
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      10/10/79      REQUIREMENTS
C      E H SCHLOSSER      LEC      10/11/79      DESIGN & CODE
C
C METHOD
C -----
C
C      DETERMINE ACTUAL/IMPLIED VALUES FOR OPTIONAL ARGUMENTS.
C      ENCODE REAL WITH MINIMUM FIELDWIDTH OF MINFLW & MAXIMUM FIELDWIDTH
C      OF 12.  MOVE ENCODED STRING TO FIRST AVAILABLE LOCATION IN OUTPUT
C      CHARACTER BUFFER AND UPDATE BUFFER POINTER.  THIS TRANSFORM PROVIDES
C      EXTENDED PASCAL-TYPE REAL ENCODING.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      UNIVAC FORTRAN V RETURN K.
C
C EXTERNAL REFERENCES
C -----
C
C      ARORET      8 DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR
C      CST4RL      8 CHARACTER STRING FOR REAL
C      INTEGER IDUP      8 INTEGER DUPLICATE (INDIRECT REFERENCE TO OPTIONAL ARG)
C
C EXCEPTIONS
C -----
C
C      1.  IF KBOUT IS SMALLER IN SIZE THAN A STANDARD CHARACTER BUFFER
C          (SUCH AS ICBUF1, ICBUF2, ETC.) THEN THE RESULTS ARE UNDEFINED.
C
C      2.  IF KBOUT HAS NOT BEEN PREVIOUSLY INITIALIZED BY CBINIT THE RESULTS
C          ARE UNDEFINED.
C
C      3.  IF THERE IS INSUFFICIENT ROOM REMAINING IN THE OUTPUT BUFFER, THEN
C          ONE OR MORE '?' WILL BE APPENDED TO THE BUFFER, IF POSSIBLE.
C
C GLOBAL DECLARATIONS

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CB4RL  
002

```

C -----
C
C      INCLUDE CDEF.LIST      & DEFINE STRUCTURE OF CHARACTER BUFFER
C      INCLUDE NULCST.LIST    & DEFINE NULL CHARACTER STRING
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER KBOUT(1)      & ARGUMENT
C      INTEGER NAROS         & NUMBER OF ACTUAL ARGUMENTS
C      INTEGER KRETN         & RETURN K VECTOR
C      INTEGER MINIFW        & MINIMUM FIELD WIDTH
C      INTEGER NDIOPR        & NUMBER OF DIGITS OF PRECISION
C      INTEGER LEADZC        & LEADING ZERO CHARACTER
C
C
C PROCEDURE
C -----
C
C DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR
C
C      CALL ARGRET(NAROS,KRETN)
C
C DETERMINE VALUES FOR OPTIONAL ARGUMENTS
C
C      MINIFW=12
C      IF(NAROS.GE.3) MINIFW=IDUP(MINIFW)
C      NDIOPR=2
C      IF(NAROS.GE.4) NDIOPR=IDUP(NDIOPR)
C      LEADZC=' '
C      IF(NAROS.GE.5) LEADZC=IDUP(LEADZC)
C
C
C CHECK BUFFER POINTERS
C
C      IF(KBOUT(CBNUL).NE.NULCST) GO TO 900
C      IF(KBOUT(CBLOC).LT.0) GO TO 900
C      IF(KBOUT(CBEND).GT.255) GO TO 900
C
C
C COMPUTE MAXIMUM LENGTH TO APPEND TO BUFFER, IF ANY
C
C      LENMAX=MIN0(KBOUT(CBEND)-KBOUT(CBLOC),MAX0(MINIFW,12))
C      IF(LENMAX.LE.0) GO TO 900
C
C
C ENCODE REAL AT FIRST AVAILABLE LOCATION IN BUFFER
C
C      CALL CSTR4L(KBOUT,(KBOUT(CBLOC)+1),(LENMAX),
C      REAL,MINIFW,NDIOPR,LEADZC)
C
C
C UPDATE BUFFER LOC TO POINT TO LAST CHARACTER NOW IN BUFFER

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CS4RL  
003

```
C      KBOUT(CBLOC)=LENCST(KBOUT,(KBOUT(CBLOC)+LENMAX))
C
C
C DONE
C
  900 RETURN KRETN
      END
```

```

.      SUBROUTINE CLRQWD  & CLEAR QUARTER WORD MODE
.      -----
.
.      HISTORY
.      -----
.
.      E M SCHLOSSER      LEC      07/12/70      ORIGINAL CODE
.
.      METHOD
.      -----
.
.      EXECUTIVE REQUEST.
.
.      MACHINE-DEPENDENT CODE
.      -----
.
.      WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS UNDER THE
.      EXEC-8 OPERATING SYSTEM.
.
.      EXTERNAL REFERENCES
.      -----
.
.      ER PSRS      CHANGE BIT IN PROCESSOR STATE REGISTER
.
.      EXCEPTIONS
.      -----
.
.      1. THE COMPUTER MUST BE IN QUARTER-WORD MODE WHEN CALLING ANY TRANSFORMS
.      WHICH PERFORM BYTE OR NYBLE OPERATIONS (GETBYT, PUTBYT, MOVBYT, ETC.).
.
.      GLOBAL DECLARATIONS
.      -----
.
.      AXRS      . STANDARD UNIVAC 1100 REGISTER MNEMONICS
.
.      LOCAL DECLARATIONS
.      -----
.
.      S(01) . 0-BANK      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
.      NF      FORM      10.1.10.1      .
.      MASK      NF      0.0.0.1      .
.
.      PROCEDURE
.      -----
.
.      S(00) . 1-BANK      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
.      CLRQWD      LA      A0,MASK      .
.                  ER      PSRS      .

```



**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**CLRGND  
002**

**J            1.X11            .  
END**

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CST4AB  
001

```

SUBROUTINE CST4AB( 8 CHARACTER STRING FOR ASCII BYTE STRING
0 JCSBUF, 8 CHARACTER STRING
.
1 JASBUF, 8 ASCII BYTE STRING
1 NBYTES) 8 NUMBER OF BYTES TO CONVERT
      (THE SAME ACTUAL ARGUMENT MAY BE USED FOR JASBUF & JCSBUF)
-----
C
C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      07/20/78      ORIGINAL CODE
C      E M SCHLOSSER      LEMSCO   03/26/80      TRANSLATE BYTES OVER 127 TO '?'
C
C
C METHOD
C -----
C
C      TABLE LOOK-UP.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      FIELDATA TRANSLATE TABLE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      GETBYT      8 GET BYTE FROM INTERNAL BYTE STRING
C      PUTCHR      8 PUT CHARACTER INTO CHARACTER STRING
C
C EXCEPTIONS
C -----
C
C      1. JCSBUF IS UNCHANGED IF NBYTES IS LESS THAN 1.
C
C      2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER NBY      8 NUMBER OF BYTES CONVERTED
C      INTEGER IBYT      8 BYTE BEING CONVERTED
C      INTEGER JTTCH4(64) 8 TRANSLATE TABLE -- CHARACTER FOR ASCII BYTE
C

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CSTOAS  
002

```

C
C
C      0      1      2      3      4      5      6      7
C      DATA JTTCVA /
C      0 ' ', '!', '!', '!', '!', '!', '!', '!'
C      1 '!', '!', '!', '!', '!', '!', '!', '!'
C      2 '!', '!', '!', '!', '!', '!', '!', '!'
C      3 '!', '!', '!', '!', '!', '!', '!', '!'
C      4 '!', '!', '!', '!', '!', '!', '!', '!'
C      5 '!', '!', '!', '!', '!', '!', '!', '!'
C      6 '!', '!', '!', '!', '!', '!', '!', '!'
C      7 '!', '!', '!', '!', '!', '!', '!', '!'
C      8 '!', '!', '!', '!', '!', '!', '!', '!'
C      9 '!', '!', '!', '!', '!', '!', '!', '!'
C      10 '!', '!', '!', '!', '!', '!', '!', '!'
C      11 '!', '!', '!', '!', '!', '!', '!', '!'
C      12 '!', '!', '!', '!', '!', '!', '!', '!'
C      13 '!', '!', '!', '!', '!', '!', '!', '!'
C      14 '!', '!', '!', '!', '!', '!', '!', '!'
C      15 '!', '!', '!', '!', '!', '!', '!', '!'
C      16 '!', '!', '!', '!', '!', '!', '!', '!'
C      17 '!', '!', '!', '!', '!', '!', '!', '!'
C      18 '!', '!', '!', '!', '!', '!', '!', '!'
C      19 '!', '!', '!', '!', '!', '!', '!', '!'
C      20 '!', '!', '!', '!', '!', '!', '!', '!'
C      21 '!', '!', '!', '!', '!', '!', '!', '!'
C      22 '!', '!', '!', '!', '!', '!', '!', '!'
C      23 '!', '!', '!', '!', '!', '!', '!', '!'
C      24 '!', '!', '!', '!', '!', '!', '!', '!'
C      25 '!', '!', '!', '!', '!', '!', '!', '!'
C      26 '!', '!', '!', '!', '!', '!', '!', '!'
C      27 '!', '!', '!', '!', '!', '!', '!', '!'
C      28 '!', '!', '!', '!', '!', '!', '!', '!'
C      29 '!', '!', '!', '!', '!', '!', '!', '!'
C      30 '!', '!', '!', '!', '!', '!', '!', '!'
C      31 '!', '!', '!', '!', '!', '!', '!', '!'
C      32 '!', '!', '!', '!', '!', '!', '!', '!'
C      33 '!', '!', '!', '!', '!', '!', '!', '!'
C      34 '!', '!', '!', '!', '!', '!', '!', '!'
C      35 '!', '!', '!', '!', '!', '!', '!', '!'
C      36 '!', '!', '!', '!', '!', '!', '!', '!'
C      37 '!', '!', '!', '!', '!', '!', '!', '!'
C      38 '!', '!', '!', '!', '!', '!', '!', '!'
C      39 '!', '!', '!', '!', '!', '!', '!', '!'
C      40 '!', '!', '!', '!', '!', '!', '!', '!'
C      41 '!', '!', '!', '!', '!', '!', '!', '!'
C      42 '!', '!', '!', '!', '!', '!', '!', '!'
C      43 '!', '!', '!', '!', '!', '!', '!', '!'
C      44 '!', '!', '!', '!', '!', '!', '!', '!'
C      45 '!', '!', '!', '!', '!', '!', '!', '!'
C      46 '!', '!', '!', '!', '!', '!', '!', '!'
C      47 '!', '!', '!', '!', '!', '!', '!', '!'
C      48 '!', '!', '!', '!', '!', '!', '!', '!'
C      49 '!', '!', '!', '!', '!', '!', '!', '!'
C      50 '!', '!', '!', '!', '!', '!', '!', '!'
C      51 '!', '!', '!', '!', '!', '!', '!', '!'
C      52 '!', '!', '!', '!', '!', '!', '!', '!'
C      53 '!', '!', '!', '!', '!', '!', '!', '!'
C      54 '!', '!', '!', '!', '!', '!', '!', '!'
C      55 '!', '!', '!', '!', '!', '!', '!', '!'
C      56 '!', '!', '!', '!', '!', '!', '!', '!'
C      57 '!', '!', '!', '!', '!', '!', '!', '!'
C      58 '!', '!', '!', '!', '!', '!', '!', '!'
C      59 '!', '!', '!', '!', '!', '!', '!', '!'
C      60 '!', '!', '!', '!', '!', '!', '!', '!'
C      61 '!', '!', '!', '!', '!', '!', '!', '!'
C      62 '!', '!', '!', '!', '!', '!', '!', '!'
C      63 '!', '!', '!', '!', '!', '!', '!', '!'
C      64 '!', '!', '!', '!', '!', '!', '!', '!'
C      65 '!', '!', '!', '!', '!', '!', '!', '!'
C      66 '!', '!', '!', '!', '!', '!', '!', '!'
C      67 '!', '!', '!', '!', '!', '!', '!', '!'
C      68 '!', '!', '!', '!', '!', '!', '!', '!'
C      69 '!', '!', '!', '!', '!', '!', '!', '!'
C      70 '!', '!', '!', '!', '!', '!', '!', '!'
C      71 '!', '!', '!', '!', '!', '!', '!', '!'
C      72 '!', '!', '!', '!', '!', '!', '!', '!'
C      73 '!', '!', '!', '!', '!', '!', '!', '!'
C      74 '!', '!', '!', '!', '!', '!', '!', '!'
C      75 '!', '!', '!', '!', '!', '!', '!', '!'
C      76 '!', '!', '!', '!', '!', '!', '!', '!'
C      77 '!', '!', '!', '!', '!', '!', '!', '!'
C      78 '!', '!', '!', '!', '!', '!', '!', '!'
C      79 '!', '!', '!', '!', '!', '!', '!', '!'
C      80 '!', '!', '!', '!', '!', '!', '!', '!'
C      81 '!', '!', '!', '!', '!', '!', '!', '!'
C      82 '!', '!', '!', '!', '!', '!', '!', '!'
C      83 '!', '!', '!', '!', '!', '!', '!', '!'
C      84 '!', '!', '!', '!', '!', '!', '!', '!'
C      85 '!', '!', '!', '!', '!', '!', '!', '!'
C      86 '!', '!', '!', '!', '!', '!', '!', '!'
C      87 '!', '!', '!', '!', '!', '!', '!', '!'
C      88 '!', '!', '!', '!', '!', '!', '!', '!'
C      89 '!', '!', '!', '!', '!', '!', '!', '!'
C      90 '!', '!', '!', '!', '!', '!', '!', '!'
C      91 '!', '!', '!', '!', '!', '!', '!', '!'
C      92 '!', '!', '!', '!', '!', '!', '!', '!'
C      93 '!', '!', '!', '!', '!', '!', '!', '!'
C      94 '!', '!', '!', '!', '!', '!', '!', '!'
C      95 '!', '!', '!', '!', '!', '!', '!', '!'
C      96 '!', '!', '!', '!', '!', '!', '!', '!'
C      97 '!', '!', '!', '!', '!', '!', '!', '!'
C      98 '!', '!', '!', '!', '!', '!', '!', '!'
C      99 '!', '!', '!', '!', '!', '!', '!', '!'
C
C PROCEDURE
C -----
C
C CHECK FOR NULL CONVERSION
C
C      IF(NBYTES.LE.0) GO TO 900
C
C
C TRANSLATE WHILE SCANNING FROM LEFT TO RIGHT
C
C      DO 900 NBY=1,NBYTES
C          CALL GETBYT(1BYT, JASBUF.(NBY))
C          IF(1BYT.GT.127) 1BYT=63      & ASCII '?'
C          1BYT=MAX0(1BYT-32,0)      & CONVERT ALL ASCII CONTROL CHARS TO SPACE
C          IF(1BYT.GT.63) 1BYT=1BYT-32      & CONVERT LOWER TO UPPER CASE
C          CALL PUTCHR(JCSBUF.(NBY), JTTCVA(1BYT+1))
C      900 CONTINUE
C
C
C DONE
C
C      900 RETURN
C          END

```

```

SUBROUTINE CST4EB( 3 CHAR STRING FOR EBCDIC BYTE STRING (FIELDATA)
0 JCSBUF. 3 CHARACTER STRING
1
1 JESBUF. 3 EBCDIC BYTE STRING
1 NBYTES) 3 NUMBER OF BYTES TO CONVERT
      (THE SAME ACTUAL ARGUMENT MAY BE USED FOR JESBUF & JCSBUF)
-----
C
C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      07/20/78      ORIGINAL CODE
C
C
C METHOD
C -----
C
C      TABLE LOOK-UP.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      TRANSLATE TABLE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      GETBYT      3 GET BYTE FROM BYTE STRING
C      PUTCHR      3 PUT CHARACTER INTO CHARACTER STRING
C
C EXCEPTIONS
C -----
C
C      1. JCSBUF IS UNCHANGED IF NBYTES IS LESS THAN 1.
C
C      2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER NBY      3 NUMBER OF BYTES CONVERTED
C      INTEGER IBYT      3 BYTE BEING CONVERTED
C      INTEGER JTTCH4E(84) 3 CHARACTER TRANSLATE TABLE
C
C

```

C-7

**CST450**  
**002**

**R-30**

```

SUBROUTINE CST4IN( 8 CHARACTER STRING FOR INTEGER
0 KSTOUT, 8 OUTPUT CHARACTER STRING
( LOCOUT, 8 CHARACTER LOC WITHIN OUTPUT STRING WHERE SUBSTRING BEGINS
( LENOUT, 8 LENGTH (>0) OF OUTPUT SUBSTRING
.
I INTEGER, 8 INTEGER TO ENCODE
I MINFLW, 8 (OPTIONAL) MINIMUM FIELDWIDTH (LENOUT IF OMITTED)
I LOZCHR, 8 (OPTIONAL) LEADING ZERO CHARACTER (BLANK IF OMITTED)
8 8.8) 8 TO KEEP COMPILER STRAIGHT -- DON'T USE!!!
      (CHAR LOCATION COUNTED FROM 1 AT LEFT OF STRING)
-----
C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      09/20/79      REQUIREMENTS
C      E M SCHLOSSER      LEC      10/04/79      DESIGN & CODE
C
C
C METHOD
C -----
C
C      DETERMINE ACTUAL/IMPLIED VALUES FOR OPTIONAL ARGUMENTS.
C      ENCODE INTEGER INTO SUBSTRING WITH MINIMUM FIELDWIDTH OF MINFLW &
C      MAXIMUM FIELDWIDTH OF LENOUT. THIS TRANSFORM IMPLEMENTS EXTENDED
C      PASCAL-TYPE INTEGER ENCODING.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      UNIVAC FORTRAN V RETURN K.
C
C EXTERNAL REFERENCES
C -----
C
C      AR0RET      8 DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR
C      MOVCSY      8 MOVE CHARACTER STRING
C      PUTCHR      8 PUT CHARACTER INTO CHARACTER STRING
C      PUTICE      8 PUT INTEGER-CHARACTER-EQUIVALENT INTO CHARACTER STRING
C      INTEGER IDUP      8 INTEGER DUPLICATE (INDIRECT REFERENCE TO OPTIONAL ARG)
C      INTEGER ICE      8 INTEGER CHARACTER EQUIVALENT
C
C
C EXCEPTIONS
C -----
C
C      1. IF THE OUTPUT SUBSTRING LIES WHOLLY OR PARTIALLY OUTSIDE ITS STRING THE
C          RESULTS WILL BE UNDEFINED.
C
C      2. IF THE ENCODED INTEGER (INCLUDING MINUS SIGN IF NEGATIVE) REQUIRES MORE
C          THAN LENOUT CHARACTERS. THEN CST4IN WILL RETURN ALL '?' IN THE OUTPUT
C          SUBSTRING.
C

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CST4IN  
002

C 3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

C

C

C GLOBAL DECLARATIONS

C -----

C

C

NONE.

C

C

C LOCAL DECLARATIONS

C -----

C

INTEGER NAROS	% NUMBER OF ACTUAL ARGUMENTS
INTEGER KRETN	% RETURN K VECTOR
INTEGER INQREH	% INTEGER QUOTIENT REMNANT (ABSOLUTE VALUE)
INTEGER LOUTAB	% LOC OF OUTPUT SUBSTRING ABSOLUTE VALUE
INTEGER LCOD	% LOC OF CODED STRING
INTEGER LFLO	% LOC OF MINIMUM WIDTH FIELD
INTEGER LFLOAB	% LOC OF MINIMUM WIDTH FIELD ABSOLUTE VALUE
INTEGER LFIN	% COMMON FINAL LOCATION
INTEGER ICEDIG	% INTEGER-CHARACTER-EQUIVALENT OF ENCODED DIGIT
INTEGER ICEZER	% INTEGER-CHARACTER-EQUIVALENT OF ZERO CHARACTER
INTEGER LEADZC	% LEADING ZERO CHARACTER

C

C

C PROCEDURE

C -----

C

C

C DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR

C

CALL ARGRET(NAROS,KRETN)

C

C

C INITIALIZE LOCATION POINTERS

C

LFIN=LOCOUT+LENOUT-1  
LFLO=LOCOUT  
IF(NAROS.GE.9) LFLO=LFIN-10UP(MINFLW)+1  
LFLO=MAX0(LFLO,LOCOUT)

C

C

C INITIALIZE SIGN-DEPENDENT VARIABLES

C

INQREH=1ABS(INTOER)  
LOUTAB=LOCOUT  
LFLOAB=LFLO  
IF(INTOER.GE.0) GO TO 100  
LOUTAB=LOUTAB+1 % LEAVE ROOM FOR MINUS SIGN  
LFLOAB=LFLOAB+1

100 CONTINUE

C

C

C INITIALIZE ZERO CHARACTER & LEADING ZERO CHARACTER

C

ICEZER=ICE('0')

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CST41N  
003

```

      LEADZC=' '
      IF(AROS.GE.8) LEADZC=10UP(LDZCHR)
C
C
C ENCODE DIGITS FROM RIGHT TO LEFT UNTIL QUOTIENT REMNANT IS ZERO
C
      IF(LFIN.LT.LOUTAB) GO TO 300
      DO 200 LCOD=LFIN,LOUTAB,-1
        ICEDIG=MOD(INGREN,10)+ICEZER
        INGREM=INGREN/10
        CALL PUTICE(KSTOUT,(LCOD), ICEDIG)
        IF(INGREN.EQ.0) GO TO 400
      200 CONTINUE
C
C
C FLAG ENCODING OVERFLOW
C
      300 CALL MOVCS(KSTOUT,(LOCOUT),(LENOUT),
        '?(11,11,?'')
      GO TO 900
C
C
C ENCODE ANY REMAINING NON-BLANK LEADING ZERO CHARS IN MINIMUM WIDTH FIELD
C
      400 IF(LEADZC.EQ.' ') GO TO 500
      420 IF(LCOD.LE.LFLOAB) GO TO 500
        LCOD=LCOD-1
        CALL PUTCHR(KSTOUT,(LCOD), LEADZC)
        GO TO 420
C
C
C ENCODE MINUS SIGN IF INTEGER IS NEGATIVE
C
      500 IF(INTER.GE.0) GO TO 600
        LCOD=LCOD-1
        CALL PUTCHR(KSTOUT,(LCOD), '-')
C
C
C ENCODE ANY REMAINING BLANK CHARACTERS IN MINIMUM WIDTH FIELD
C
      600 IF(LCOD.LE.LFLO) GO TO 700
        LCOD=LCOD-1
        CALL PUTCHR(KSTOUT,(LCOD), ' ')
        GO TO 600
C
C
C SHIFT ENCODED STRING INTO PROPER ALIGNMENT
C
      700 CALL MOVCS(KSTOUT,(LOCOUT),(LENOUT),
        KSTOUT,(LCOD),(LFIN-LCOD+1),' ')
C
C
C DONE
C
      900 RETURN KRETN
      END

```



SUBROUTINE CST4RL( 8 CHARACTER STRING FOR REAL  
0 KSTOUT. 8 OUTPUT CHARACTER STRING  
1 LOCOUT. 8 CHARACTER LOC WITHIN OUTPUT STRING WHERE SUBSTRING BEGINS  
1 LENOUT. 8 LENGTH (10) OF OUTPUT SUBSTRING  
.  
1 REAL. 8 REAL TO ENCODE  
1 MINFLW. 8 (OPTIONAL) MINIMUM FIELDWIDTH (LENOUT IF OMITTED)  
1 NOPREC. 8 (OPTIONAL) NUMBER OF DIGITS OF PRECISION (2 IF OMITTED)  
1 LDZCHR. 8 (OPTIONAL) LEADING ZERO CHARACTER (BLANK IF OMITTED)  
8 8.8) 8 TO KEEP COMPILER STRAIGHT -- DON'T USE!!!  
(CHAR LOCATION COUNTED FROM 1 AT LEFT OF STRING)

C  
C  
C  
C  
C HISTORY

C  
C  
C E H SCHLOSSER LEC 10/10/79 REQUIREMENTS  
C E H SCHLOSSER LEC 10/11/79 DESIGN & CODE

C  
C  
C  
C  
C METHOD

C  
C  
C DETERMINE ACTUAL/IMPLIED VALUES FOR OPTIONAL ARGUMENTS.  
C ENCODE REAL INTO SUBSTRING WITH MINIMUM FIELDWIDTH OF MINFLW &  
C MAXIMUM FIELDWIDTH OF LENOUT. THIS TRANSFORM IMPLEMENTS EXTENDED  
C PASCAL-TYPE REAL ENCODING.

C  
C  
C  
C  
C MACHINE-DEPENDENT CODE

C  
C  
C UNIVAC FORTRAN V RETURN K.

C  
C  
C  
C  
C EXTERNAL REFERENCES

C  
C  
C ARORET 8 DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR  
C MOVCS 8 MOVE CHARACTER STRING  
C PUTCHR 8 PUT CHARACTER INTO CHARACTER STRING  
C PUTICE 8 PUT INTEGER-CHARACTER-EQUIVALENT INTO CHARACTER STRING  
C INTEGER IDUP 8 INTEGER DUPLICATE (INDIRECT REFERENCE TO OPTIONAL ARG)  
C INTEGER ICE 8 INTEGER CHARACTER EQUIVALENT

C  
C  
C  
C  
C EXCEPTIONS

- C  
C  
C 1. IF THE OUTPUT SUBSTRING LIES WHOLLY OR PARTIALLY OUTSIDE ITS STRING THE  
C RESULTS WILL BE UNDEFINED.  
C  
C 2. IF THE ENCODED REAL (INCLUDING MINUS SIGN IF NEGATIVE) REQUIRES MORE  
C THAN LENOUT CHARACTERS. THEN CST4RL WILL RETURN ALL '?' IN THE OUTPUT  
C SUBSTRING.

```

C
C      3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER NAROS      & NUMBER OF ACTUAL ARGUMENTS
C      INTEGER KRETN      & RETURN K VECTOR
C      REAL QUOREM        & REAL QUOTIENT REMNANT (ABSOLUTE VALUE)
C      INTEGER LOUTAB     & LOC OF OUTPUT SUBSTRING ABSOLUTE VALUE
C      INTEGER LCOD       & LOC OF CODED STRING
C      INTEGER LFLD       & LOC OF MINIMUM WIDTH FIELD
C      INTEGER LFLOAB     & LOC OF MINIMUM WIDTH FIELD ABSOLUTE VALUE
C      INTEGER LDOT       & LOC OF DOT (DECIMAL POINT)
C      INTEGER LFIN       & COMMON FINAL LOCATION
C      INTEGER ICEDIO     & INTEGER-CHARACTER-EQUIVALENT OF ENCODED DIOIT
C      INTEGER ICEZER     & INTEGER-CHARACTER-EQUIVALENT OF ZERO CHARACTER
C      INTEGER NDIOPR     & NUMBER OF DIGITS OF PRECISION
C      INTEGER LEADZC     & LEADING ZERO CHARACTER
C
C
C C PROCEDURE
C -----
C
C
C DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR
C
C      CALL ARORET(NAROS,KRETN)
C
C
C C INITIALIZE LOCATION POINTERS & DOT
C
C      LFIN=LOCOUT+LENOUT-1
C      LFLD=LOCOUT
C      IF(NAROS.GE.5) LFLD=LFIN-IDUP(MINFLW)+1
C      LFLD=MAX0(LFLD,LOCOUT)
C      NDIOPR=2
C      IF(NAROS.GE.6) NDIOPR=MAX0(IDUP(NOPREC),0)
C      LDOT=LFIN-NDIOPR
C      CALL PUTCHR(KSTOUT,(LDOT),  ' ')
C
C
C C INITIALIZE SIGN-DEPENDENT VARIABLES
C
C      QUOREM=ABS(REAL)*FLOAT(10**NDIOPR)+0.5 & ABSOLUTE VALUE. SCALED & ROUNDED
C      LOUTAB=LOCOUT
C      LFLOAB=LFLD
C      IF(REAL.GE.0) GO TO 100
C      LOUTAB=LOUTAB+1 & LEAVE ROOM FOR MINUS SIGN

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CST4RL  
003

```

        LFLDAB=LFLDAB+1
100 CONTINUE
C
C
C INITIALIZE ZERO CHARACTER & LEADING ZERO CHARACTER
C
        ICEZER=ICE('0')
        LEADZC=' '
        IF(MAROS.OE.7) LEADZC=IDUP(LOZCHR)
C
C
C ENCODE DIGITS FROM RIGHT TO LEFT UNTIL QUOTIENT REMNANT < 1.0
C
        IF(LFIN.LT.LOUTAB) GO TO 300
        DO 200 LCOO=LFIN.LOUTAB,-1
            IF(LCOO.EQ.LDOT) GO TO 180
            ICEDIG=MIN(9,IFIX(AMOD(QUOREM,10.))) + ICEZER
            QUOREM=QUOREM/10.
            CALL PUTICE(KSTOUT,(LCOO), ICEDIG)
180        IF((QUOREM.LT.1.0).AND.(LCOO.LE.LDOT)) GO TO 400
200        CONTINUE
C
C
C FLAG ENCODING OVERFLOW
C
300 CALL MOVCST(KSTOUT,(LOCOUT),(LENOUT),
    - '7',(1),(1),'7')
        GO TO 900
C
C
C ENCODE ANY REMAINING NON-BLANK LEADING ZERO CHARS IN MINIMUM WIDTH FIELD
C
400 IF(LEADZC.EQ.' ') GO TO 500
420 IF(LCOO.LE.LFLDAB) GO TO 500
        LCOO=LCOO-1
        CALL PUTCHR(KSTOUT,(LCOO), LEADZC)
        GO TO 420
C
C
C ENCODE MINUS SIGN IF INTEGER IS NEGATIVE
C
500 IF(REAL.OE.0) GO TO 600
        LCOO=LCOO-1
        CALL PUTCHR(KSTOUT,(LCOO), '-')
C
C
C ENCODE ANY REMAINING BLANK CHARACTERS IN MINIMUM WIDTH FIELD
C
600 IF(LCOO.LE.LFLD) GO TO 700
        LCOO=LCOO-1
        CALL PUTCHR(KSTOUT,(LCOO), ' ')
        GO TO 600
C
C
C SHIFT ENCODED STRING INTO PROPER ALIGNMENT
C

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CST4RL  
004

```
      700 CALL MOVEST(KSTOUT,(LOCOUT),(LENOUT).  
      *          KSTOUT,(LCOD),(LFIN-LCOD+1),* *)  
C  
C  
C DONE  
C  
      900 RETURN KRETN  
      END
```

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**CURBST  
001**

**(NOT IMPLEMENTED)**

```

      DOUBLE PRECISION FUNCTION
      0      0 VARIABLE-LENGTH STRING (<= 0 CHARS)
      = CBS4CS!
      I KSTIN. 0 INPUT CHARACTER STRING
      ( LOCIN. 0 (OPTIONAL) LOC IN KSTIN WHERE SUBSTRING BEGINS (1 IF OMITTED)
      ( LENIN. 0 (OPTIONAL) LENGTH & DIRECTION OF INPUT SUBSTRING (0 IF OMITTED)
      C      (POSITIVE DIRECTION = LEFT TO RIGHT)
      C      (CHAR LOCATIONS COUNTED FROM 1 AT LEFT OF STRING)
      S      S.S) 0 TO KEEP COMPILER STRAIGHT -- DON'T USE!!!
      -----
      C
      C
      C
      C HISTORY
      C -----
      C      E M SCHLOSSER      LEC      10/09/79      REQUIREMENTS
      C      E M SCHLOSSER      LEC      10/15/79      DESIGN & CODE
      C
      C
      C METHOD
      C -----
      C
      C      DETERMINE ACTUAL/IMPLIED VALUES FOR OPTIONAL ARGUMENTS.
      C      MOVE INPUT SUBSTRING TO OUTPUT VARIABLE-LENGTH STRING WITH MAXIMUM
      C      LENGTH OF 0 CHARACTERS.
      C
      C
      C MACHINE-DEPENDENT CODE
      C -----
      C
      C      UNIVAC FORTRAN V RETURN K.
      C
      C
      C EXTERNAL REFERENCES
      C -----
      C
      C      ARDRET      0 DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR
      C      MOVCS1      0 MOVE CHARACTER STRING
      C      INTEGER IDUP      0 INTEGER DUPLICATE (INDIRECT REFERENCE TO OPTIONAL ARG)
      C
      C
      C EXCEPTIONS
      C -----
      C
      C      1. IF CBS4.S IS NOT DECLARED DOUBLE PRECISION IN THE INVOKING ROUTINE.
      C          THE RESULTS WILL BE UNDEFINED.
      C
      C      2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
      C
      C
      C GLOBAL DECLARATIONS
      C -----
      C
      C      INCLUDE NULCHR.LIST      0 DEFINE NULL CHARACTER
      C
      C

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CS54CS  
888

C LOCAL DECLARATIONS

C -----

C

INTEGER NARG	% NUMBER OF ACTUAL ARGUMENTS
INTEGER KRET	% RETURN K VECTOR
INTEGER LOCIN	% CHAR LOC WITHIN INPUT STRING WHERE SUBSTRING BEGINS
INTEGER LENIN	% LENGTH & DIRECTION OF INPUT SUBSTRING

C

C

C PROCEDURE

C -----

C

C

C DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR

C

CALL ARGRET(NARG,KRET)

C

C

C DETERMINE VALUES FOR OPTIONAL ARGUMENTS

C

LOCIN=1  
IF(NARG.GE.2) LOCIN=1DUP(LOCIN)  
LENIN=+0  
IF(NARG.GE.3) LENIN=1DUP(LENIN)  
LENIN=MAX0(-0,MIN0(+0,LENIN))

C

C

C MOVE IT. TERMINATED WITH NULCHR

C

CALL MOVST(CS54CS,(1),(1ABS(LENIN)+1),  
KSTIN,LOCIN,LENIN,NULCHR)

C

C

C DONE

C

RETURN KRET  
END

```

      DOUBLE PRECISION FUNCTION
      @      @ VARIABLE-LENGTH STRING (12 CHARPS) CONTAINING ENCODED INTEGER
      @ CBS4IN(
      @ I INTEGER. @ INTEGER TO ENCODE
      @ I MINFLW. @ (OPTIONAL) MINIMUM FIELDWIDTH (0 IF OMITTED)
      @ I LDZCHR. @ (OPTIONAL) LEADING ZERO CHARACTER (BLANK IF OMITTED)
      @ @ @ @ @ TO KEEP COMPILER STRAIGHT -- DON'T USE!!!
      @-----
C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      09/20/79      REQUIREMENTS
C      E M SCHLOSSER      LEC      10/04/79      DESIGN & CODE
C
C
C METHOL
C -----
C
C      DETERMINE ACTUAL/IMPLIED VALUES FOR OPTIONAL ARGUMENTS.
C      ENCODE INTEGER INTO VARIABLE-LENGTH CHARACTER STRING WITH MINIMUM
C      FIELDWIDTH OF MINFLW & MAXIMUM FIELDWIDTH OF @. THIS FUNCTION PROVIDES
C      EXTENDED PASCAL-TYPE INTEGER ENCODING.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      UNIVAC FORTRAN V RETURN K.
C
C
C EXTERNAL REFERENCES
C -----
C
C      AROREY      @ DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR
C      PUTCHR      @ PUT CHARACTER INTO CHARACTER STRING
C      CST4IN      @ CHARACTER STRING FOR INTEGER
C      INTEGER IDUP      @ INTEGER DUPLICATE (INDIRECT REFERENCE TO OPTIONAL ARG)
C      INTEGER LENGST      @ LENGTH OF CHARACTER STRING
C
C
C EXCEPTIONS
C -----
C
C      1. IF CBS4IN IS NOT DECLARED DOUBLE PRECISION IN THE INVOKING ROUTINE,
C          THE RESULTS WILL BE UNDEFINED.
C
C      2. IF THE ENCODED INTEGER (INCLUDING MINUS SIGN IF NEGATIVE) REQUIRES
C          MORE THAN @ CHARACTERS, THEN CBS4IN WILL RETURN A STRING OF '?'.
C
C      3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
C
C
C GLOBAL DECLARATIONS
C -----

```



DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CBS4IN  
002

```

C      INCLUDE NULCHR.LIST      & DEFINE NULL CHARACTER
C
C
C LOCAL DECLARATIONS
C -----
C      INTEGER NARGOS           & NUMBER OF ACTUAL ARGUMENTS
C      INTEGER KRETN           & RETURN K VECTOR
C      INTEGER MINIFW          & MINIMUM FIELDWIDTH
C      INTEGER LEADZC           & LEADING ZERO CHARACTER
C
C
C PROCEDURE
C -----
C
C DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR
C
C      CALL ARGRET(NARGOS,KRETN)
C
C
C DETERMINE VALUES FOR OPTIONAL ARGUMENTS
C
C      MINIFW=0
C      IF(NARGOS.GE.2) MINIFW=1DUP(MINIFW)
C      LEADZC=' '
C      IF(NARGOS.GE.3) LEADZC=1DUP(LEADZC)
C
C
C ENCODE IT. TERMINATED WITH NULCHR
C
C      CALL CST4IN(CBS4IN,(1),(0), INTEGER,MINIFW,LEADZC)
C      CALL PUTCHR(CBS4IN,(LENCST(CBS4IN,0)+1), NULCHR)
C
C
C DONE
C
C      RETURN KRETN
C      END

```

```

      DOUBLE PRECISION FUNCTION
      0      8 VARIABLE-LENGTH STRING (<= 8 CHARS) CONTAINING ENCODED REAL
      * CS4RL(
      I REAL.      8 REAL TO ENCODE
      I MINFLW.    8 (OPTIONAL) MINIMUM FIELDWIDTH (8 IF OMITTED)
      I NOPREC.    8 (OPTIONAL) NUMBER OF DIGITS OF PRECISION (2 IF OMITTED)
      I LOZCHR.    8 (OPTIONAL) LEADING ZERO CHARACTER (BLANK IF OMITTED)
      S      S.S)  8 TO KEEP COMPILER STRAIGHT -- DON'T USE!!!
      -----
C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      10/04/79      REQUIREMENTS
C      E H SCHLOSSER      LEC      10/12/79      DESIGN & CODE
C
C
C METHOD
C -----
C
C      DETERMINE ACTUAL/IMPLIED VALUES FOR OPTIONAL ARGUMENTS.
C      ENCODE REAL INTO VARIABLE-LENGTH CHARACTER STRING WITH MINIMUM
C      FIELDWIDTH OF MINFLW & MAXIMUM FIELDWIDTH OF 8. THIS FUNCTION PROVIDES
C      EXTENDED PASCAL-TYPE REAL ENCODING.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      UNIVAC FORTRAN V RETURN K.
C
C
C EXTERNAL REFERENCES
C -----
C
C      ARGRET      8 DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR
C      PUTCHR      8 PUT CHARACTER INTO CHARACTER STRING
C      CST4RL      8 CHARACTER STRING FOR REAL
C      INTEGER IDUP      8 INTEGER DUPLICATE (INDIRECT REFERENCE TO OPTIONAL ARG)
C      INTEGER LENCST    8 LENGTH OF CHARACTER STRING
C
C
C EXCEPTIONS
C -----
C
C      1. IF CS4RL IS NOT DECLARED DOUBLE PRECISION IN THE INVOKING ROUTINE.
C          THE RESULTS WILL BE UNDEFINED.
C
C      2. IF THE ENCODED REAL (INCLUDING MINUS SIGN IF NEGATIVE) REQUIRES
C          MORE THAN 8 CHARACTERS. THEN CS4RL WILL RETURN A STRING OF '?'.
C
C      3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
C
C GLOBAL DECLARATIONS

```

DAM PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CBS4RL  
002

```

C -----
C
C      INCLUDE NULCHR.LIST      & DEFINE NULL CHARACTER
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER NARGO          & NUMBER OF ACTUAL ARGUMENTS
C      INTEGER KRETN          & RETURN K VECTOR
C      INTEGER MINIFW         & MINIMUM FIELDWIDTH
C      INTEGER NOIGPR         & NUMBER OF DIGITS OF PRECISION
C      INTEGER LEADZC         & LEADING ZERO CHARACTER
C
C
C PROCEDURE
C -----
C
C DETERMINE NUMBER OF ACTUAL ARGUMENTS & RETURN K VECTOR
C
C      CALL ARGRET(NARGO,KRETN)
C
C DETERMINE VALUES FOR OPTIONAL ARGUMENTS
C
C      MINIFW=8
C      IF(NARGO.GE.2) MINIFW=IDUP(MINIFW)
C      NOIGPR=2
C      IF(NARGO.GE.3) NOIGPR=IDUP(NOIGPR)
C      LEADZC=' '
C      IF(NARGO.GE.4) LEADZC=IDUP(LEADZC)
C
C
C ENCODE IT. TERMINATED WITH NULCHR
C
C      CALL CST4RL(CBS4RL.(1).(8), REAL.MINIFW.NOIGPR.LEADZC)
C      CALL PUTCHR(CBS4RL.(LENCST(CBS4RL.8)+1), NULCHR)
C
C
C DONE
C
C      RETURN KRETN
C      END

```

```

SUBROUTINE DCODE( 3 DECODE NUMERIC CHARACTER STRING
0 INTGR. 3 INTEGER REPRESENTATION OF DECODED CHAR STRING (TRUNCATED)
0 REAL. 3 REAL REPRESENTATION OF DECODED CHAR STRING
0 KODTYP. 3 CODE TYPE:
C 'IN' INTEGER [(SIGN)<DIGIT(S)>
C 'RL' REAL [(SIGN)<DIGIT(S)>].[<DIGIT(S)>]
C 'FR' FRACTION [(SIGN)<DIGIT(S)>]/<DIGIT(S)>
C 'SX' SEX'Y [(SIGN)<DIGIT(S)>]:<2DIGITS>[.[:<2DIGITS>]1.[<DIGIT(S)>]1]
C 'ERR' ERROR
*
I KODBUF. 3 CHARACTER STRING BUFFER CONTAINING ENCODED NUMBER
I KODLOC. 3 LOCATION OF ENCODED NUMBER
I KODLEN) 3 LENGTH OF ENCODED NUMBER
-----
C
C
C
C HISTORY
C -----
C
C E H SCHLOSSER LEC 02/10/75 ORIG CODE IN RITON/RITORL/RITOM
C E H SCHLOSSER LEC 02/13/79 REVISE & COMBINE
C E H SCHLOSSER LEC 07/11/79 ALLOW LEADING BLANKS
C
C
C
C METHOD
C -----
C
C STATE TRANSITIONS.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C GETICE 3 GET INTEGER-CHARACTER-EQUIVALENT FROM CHARACTER STRING
C INTEGER LCHRNE 3 LOCATION OF FIRST CHARACTER NOT EQUAL TO
C INTEGER ICE 3 INTEGER-CHARACTER-EQUIVALENT FROM CHARACTER
C
C
C EXCEPTIONS
C -----
C
C 1. EMBEDDED/TRAILING BLANKS ARE NOT ALLOWED.
C
C 2. THE WHOLE NUMBER PORTION MAY NOT CONTAIN MORE THAN 9 NUMERIC DIGITS.
C
C 3. WHEN KODTYP IS 'ERR'. THEN THE CONTENTS OF INTGR & REAL ARE UNDEFINED.
C
C
C GLOBAL DECLARATIONS
C -----

```

DAM PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

DCODE  
002

```

C
C      NONE.
C
C
C
C LOCAL DECLARATIONS
C -----
C
      INTEGER INSIGN      & SIGN ( -1 OR +1 )
      INTEGER NDENOM      & DENOMINATOR OF FRACTION
      REAL SCALE          & SCALE FACTOR FOR DIGIT AFTER DECIMAL/SEX'Y POINT
      INTEGER LOCOET      & LOCATION OF CHARACTER BEING ANALYZED
      INTEGER LOCEND      & LOCATION OF LAST CHARACTER TO BE ANALYZED
      INTEGER NICE        & I-C-E OF CHARACTER BEING ANALYZED
      INTEGER NCHAR       & NUMBER OF CHARS ANALYZED IN CURRENT STATE
      INTEGER NSXPL       & NUMBER OF SEXAGENARY PLACES ANALYZED IN CURRENT STATE
C
C
C PROCEDURE
C -----
C
C
C INITIALIZE
C
      INSIGN=+1
      INTOER=0
      REAL=0.0
      SCALE=1.0
      NDENOM=0
      KODTYP='ERR'
      LOCOET=LCHRNE(KODBUF,KODLOC,KODLEN,' ')
      IF(LOCOET.LE.0) GO TO 900      & INVALID IF ALL BLANKS
      LOCOET=LOCOET-1
      IF(KODLEN.LE.0) GO TO 900
      LOCEND=KODLOC+KODLEN-1
C
C
C INITIAL STATE (GET FEWER THAN 2 SIGNS)
C
      DO 130 NCHAR=1,2
        LOCOET=LOCOET+1
        IF(LOCOET.GT.LOCEND) GO TO 900
        CALL GETICE(NICE, KODBUF,(LOCOET))
        IF((NICE.EQ.ICE('0')).AND.(NICE.LE.ICE('9')) GO TO 200
        IF(NICE.EQ.ICE('.')) GO TO 300
        IF(NICE.EQ.ICE('-')) INSIGN=-1
        IF((NICE.NE.ICE('+')).AND.(NICE.NE.ICE('-')) GO TO 900
      130 CONTINUE
      GO TO 900
C
C
C
C INTEGER STATE (GET FEWER THAN 10 DIGITS BEFORE DECIMAL POINT)
C
      200 DO 230 NCHAR=1,9
        INTOER=INTOER*10 + (NICE-ICE('0'))
        LOCOET=LOCOET+1
        IF(LOCOET.GT.LOCEND) GO TO 290

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

OCODE  
003

```

        CALL GETICE(NICE, KODBUF,(LOCGET))
        IF(NICE.EQ.ICE('.')) GO TO 300
        IF(NICE.EQ.ICE('/')) GO TO 400
        IF(NICE.EQ.ICE(':')) GO TO 500
        IF((NICE.LT.ICE('0')).OR.(NICE.GT.ICE('9')) GO TO 900
230 CONTINUE
    GO TO 900
C
C
C FINALIZE INTEGER STATE
C
290 KODTYP='IN'
    REAL=FLOAT(INTEGER)
    GO TO 900
C
C
C REAL STATE (GET ANY NUMBER OF DIGITS AFTER DECIMAL POINT)
C
300 LOCGET=LOCGET+1
    IF(LOCGET.GT.LOCEND) GO TO 390
    CALL GETICE(NICE, KODBUF,(LOCGET))
    IF((NICE.LT.ICE('0')).OR.(NICE.GT.ICE('9')) GO TO 900
    SCALE=SCALE*0.1
    REAL=REAL + FLOAT(NICE-ICE('0'))*SCALE
    GO TO 300
C
C
C FINALIZE REAL STATE
C
390 KODTYP='RL'
    REAL=FLOAT(INTEGER)+REAL
    GO TO 900
C
C
C FRACTION STATE (GET FEWER THAN 10 DIGITS IN DENOMINATOR)
C
400 GO 430 NCHR=1.10
    LOCGET=LOCGET+1
    IF(LOCGET.GT.LOCEND) GO TO 490
    CALL GETICE(NICE, KODBUF,(LOCGET))
    IF((NICE.LT.ICE('0')).OR.(NICE.GT.ICE('9')) GO TO 900
    NDENOM=NDENOM*10 + (NICE-ICE('0'))
430 CONTINUE
    GO TO 900
C
C
C FINALIZE FRACTION STATE
C
490 KODTYP='FR'
    REAL=FLOAT(INTEGER)/FLOAT(NDENOM)
    INTEGER=IFIX(REAL)
    GO TO 900
C
C
C SEXADECARY STATE (GET FEWER THAN 3 SEX'Y PLACES AFTER 1ST SEX'Y POINT)
C

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

OCODE  
004

```

500 DO 530 NSXPL=1.3
      LOCOET=LOCOET+1
      IF(LOCOET.GT.LOCEND) GO TO 500
      CALL GETICE(NICE, KODBUF.(LOCOET))
      IF((NICE.LT.ICE('0')).OR.(NICE.GT.ICE('5')) GO TO 900
      SCALE=SCALE/5.0
      REAL=REAL + FLOAT(NICE-ICE('0'))*SCALE
      LOCOET=LOCOET+1
      IF(LOCOET.GT.LOCEND) GO TO 900
      CALL GETICE(NICE, KODBUF.(LOCOET))
      IF((NICE.LT.ICE('0')).OR.(NICE.GT.ICE('9')) GO TO 900
      SCALE=SCALE*0.1
      REAL=REAL + FLOAT(NICE-ICE('0'))*SCALE
      LOCOET=LOCOET+1
      IF(LOCOET.GT.LOCEND) GO TO 590
      CALL GETICE(NICE, KODBUF.(LOCOET))
      IF(NICE.EQ.ICE('.')) GO TO 560
      IF(NICE.NE.ICE(':')) GO TO 900
530 CONTINUE
      GO TO 900

C
C
C SEXAGENARY DECIMAL STATE (GET ANY NUMBER OF DIGITS AFTER DECIMAL POINT)
C
560 LOCOET=LOCOET+1
      IF(LOCOET.GT.LOCEND) GO TO 590
      CALL GETICE(NICE, KODBUF.(LOCOET))
      IF((NICE.LT.ICE('0')).OR.(NICE.GT.ICE('9')) GO TO 900
      SCALE=SCALE*0.1
      REAL=REAL + FLOAT(NICE-ICE('0'))*SCALE
      GO TO 560

C
C
C FINALIZE SEXAGENARY STATE
C
590 KODTYP='SX'
      REAL=REAL*FLOAT(INTOER)
      GO TO 900

C
C
C FINAL STATE (ASSIGN SIGN)
C
900 INTOER=ISIGN(INTOER,INSIGN)
      REAL=SIGN(REAL,FLOAT(INSIGN))
      RETURN
      END

```

```

SUBROUTINE ED4AS( 3 EBCDIC BYTE STRING FOR ASCII BYTE STRING
0 JEBBUF, 3 EBCDIC BYTE STRING
1 JASBUF, 3 ASCII BYTE STRING
1 NBYTES) 3 NUMBER OF BYTES TO CONVERT
      (THE SAME ACTUAL ARGUMENT MAY BE USED FOR JASBUF & JEBBUF)
-----
C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      07/06/78      REQUIREMENTS
C      E M SCHLOSSER      LEMSCO   01/21/80      DESIGN & CODE
C
C
C METHOD
C -----
C
C      TABLE LOOK-UP.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      GETBYT      3 GET BYTE FROM INTERNAL BYTE STRING
C      PUTBYT      3 PUT BYTE INTO INTERNAL BYTE STRING
C
C
C EXCEPTIONS
C -----
C
C      1. JEBBUF IS UNCHANGED IF NBYTES IS LESS THAN 1.
C
C      2. ASCII BYTES WITH A NUMERIC VALUE GREATER THAN 127 ARE TRANSLATED
C         TO THE EBCDIC CHARACTER CORRESPONDING TO ASCII 127.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER NBY      3 NUMBER OF BYTES CONVERTED
C      INTEGER IBYT     3 BYTE BEING CONVERTED
C      INTEGER JTTE4A(128) 3 TRANSLATE TABLE -- EBCDIC FOR ASCII
  
```



DAM PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CS4AS  
002

```

C
C
C      0      1      2      3      4      5      6      7
C      DATA (JTTE4A(N),N=1,32) /      8 CONTROL CHARACTERS
C      NUL SOH STX ETX EOT ENQ ACK BEL
C      0 000. 001. 002. 003. 005. 045. 046. 047.
C      BS  HT  LF  VT  FF  CR  SO  SI
C      1 022. 005. 037. 011. 012. 013. 014. 015.
C      DLE DC1 DC2 DC3 DC4 NAK SYN ETB
C      2 016. 017. 018. 019. 060. 061. 050. 030.
C      CAN EM SUB ESC FS OS RS US
C      3 024. 025. 063. 039. 020. 029. 030. 031/      9 RS IS ALSO ESCDIC 05377
C
C      0      1      2      3      4      5      6      7
C      DATA (JTTE4A(N),N=33,64) /      8 PUNCTUATION & NUMBERS
C      SP  !  "  #  $  %  &  '
C      4 064. 090. 127. 123. 091. 108. 080. 125.
C      ( ) * + , - . /
C      5 077. 093. 092. 078. 107. 096. 075. 097.
C      0 1 2 3 4 5 6 7
C      6 240. 241. 242. 243. 244. 245. 246. 247.
C      8 9 : ; < = > ?
C      7 248. 249. 122. 094. 076. 126. 110. 111/
C
C      0      1      2      3      4      5      6      7
C      DATA (JTTE4A(N),N=65,96) /      8 UPPER CASE
C      A B C D E F G
C      8 124. 193. 194. 195. 196. 197. 198. 199.
C      H I J K L M N O
C      9 200. 201. 209. 210. 211. 212. 213. 214.
C      P Q R S T U V W
C      0 215. 216. 217. 226. 227. 228. 229. 230.
C      X Y Z [ \ ] ^ _
C      1 231. 232. 233. 173. 224. 189. 095. 109/
C
C      0      1      2      3      4      5      6      7
C      DATA (JTTE4A(N),N=97,128) /      8 LOWER CASE
C      ACCENT A B C D E F G
C      2 121. 129. 130. 131. 132. 133. 134. 135.
C      H I J K L M N O
C      3 136. 137. 145. 146. 147. 148. 149. 150.
C      P Q R S T U V W
C      4 151. 152. 153. 162. 163. 164. 165. 166.
C      X Y Z -I OR 1- TILD DEL
C      5 167. 168. 169. 192. 079. 200. 161. 007/
C
C
C PROCEDURE
C -----
C
C CHECK FOR NULL CONVERSION
C
C      IF(INBYTS.LE.0) GO TO 900
C
C
C TRANSLATE WHILE SCANNING FROM LEFT TO RIGHT

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CHAS  
003

```
C
    DO 900 NBY=1,NBYTS
        CALL GETBYT(Ibyt,  JASBUF.(NBY))
        Ibyt=MIN(Ibyt,127)
        CALL PUTBYT(JESBUF.(NBY),  JTE4A(Ibyt+1))
    900 CONTINUE
C
C
C DONE
C
    900 RETURN
    END
```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

ED4CST  
001

```

SUBROUTINE ED4CST( 8 EBCDIC BYTE STRING FOR CHAR STRING (FIELDATA)
0 JESBUF. 8 EBCDIC BYTE STRING
.
1 JCSBUF. 8 INTERNAL CHARACTER STRING
1 NCHARS) 8 NUMBER OF CHARACTERS TO CONVERT
      (THE SAME ACTUAL ARGUMENT MAY BE USED FOR JCSBUF & JESBUF)
-----
C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      07/20/70      ORIGINAL CODE
C
C
C METHOD
C -----
C
C      TABLE LOOK-UP.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      TRANSLATE TABLE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      GETICE      8 GET I-C-E FROM CHARACTER STRING
C      PUTBYT      8 PUT BYTE INTO BYTE STRING
C
C
C EXCEPTIONS
C -----
C
C      1. JESBUF IS UNCHANGED IF NCHARS IS LESS THAN 1.
C
C      2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER NCH          8 NUMBER OF CHARACTERS CONVERTED
C      INTEGER NICE        8 I-C-E OF CHARACTER BEING CONVERTED
C      INTEGER JTTE4C(84)  8 TRANSLATE TABLE -- EBCDIC BYTE FOR CHARACTER
C
C

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

CS4CS7  
082

```

C
C      0 1 2 3 4 5 6 7
C      DATA JTTE4C /
C      0 124. 74. 90.123. 79. 84.193.194.
C
C      C D E F G H I J
C      1 195.196.197.198.199.200.201.209.
C
C      K L M N O P Q R
C      2 210.211.212.213.214.215.216.217.
C
C      S T U V W X Y Z
C      3 226.227.228.229.230.231.232.233.
C
C      ) - * < = > 6 5
C      4 93. 96. 78. 76.126.110. 80. 91.
C
C      . / 8 : ; ? _ ! , \
C      5 92. 77.108.122.111. 79.107. 95.
C
C      0 1 2 3 4 5 6 7
C      6 240.241.242.243.244.245.246.247.
C
C      8 9 . : / . #
C      7 248.249.125. 94. 97. 75.127.109/
C
C
C PROCEDURE
C -----
C
C
C CHECK FOR NULL CONVERSION
C
C      IF(NCHARS.LE.0) GO TO 900
C
C
C TRANSLATE WHILE SCANNING FROM RIGHT TO LEFT
C
C      DO 900 NCH=NCHARS.1.-1
C          CALL GETICE(NICE. JCSBUF.NCH)
C          CALL PUTBYT(JESBUF.(NCH). JTTE4C(NICE+1))
C 900 CONTINUE
C
C
C DONE
C
C 900 RETURN
C      END

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

FILED  
001

(NOT IMPLEMENTED)

```

SUBROUTINE GETBYT( I GET NON-MEO INTEGER FROM BYTE IN BYTE STRING
@ IBYT,      @ UNPACKED BYTE (RIGHT-ALIGNED & ZERO-FILLED)
.
.
I IBYSTR,    @ INTERNAL BYTE STRING
I IBYLOC)    @ BYTE LOCATION WITHIN STRING (COUNTING FROM 1 AT LEFT)
-----
.
.
HISTORY
-----
.
.
E M SCHLOSSER      LEC      07/07/70      ORIGINAL CODE
.
.
METHOD
-----
.
.
CONVERT STRING-RELATIVE BYTE LOCATION TO MACHINE-DEPENDENT
LOCATION AS FOLLOWS:
.
.
WORD NUMBER = 3* MOST SIGNIFICANT BITS
QUARTER-WORD LOCATION WITHIN WORD = 2 LEAST SIGNIFICANT BITS
.
.
MACHINE-DEPENDENT CODE
-----
.
.
WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 9-BIT
QUARTER WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.
BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES.
DIFFERENT COMPILERS (EO., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.
.
.
EXTERNAL REFERENCES
-----
.
.
NONE
.
.
EXCEPTIONS
-----
.
.
1. THE COMPUTER MUST BE IN QUARTER-WORD MODE. (ASSEMBLE WITH BASH.F
OR COLLECT WITH SHAP.F OR INSURE THAT SETQWD WAS CALLED PREVIOUSLY.)
.
.
2. IBYT IS UNCHANGED IF IBYLOC IS LESS THAN 1.
.
.
3. THE RESULT IS UNDEFINED IF IBYSTR DOES NOT CONTAIN VALID BYTE VALUES
(IN THE RANGE BETWEEN 0 AND +255).
.
.
4. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
.
.
GLOBAL DECLARATIONS
-----

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

GETBYT  
00R

AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

.  
.  
. LOCAL DECLARATIONS  
-----

. NONE  
.

. PROCEDURE  
-----

2(00) . 1-BANK

GE/BYT\*

LA  
AA.XU  
JN  
DSL  
SSL  
AA  
EX  
SA  
J

. BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY

A1.\*2.X11 . BYTE LOCATION, COUNTING FROM 1  
A1.-1 . BYTE LOCATION, COUNTING FROM 0  
A1.RETURN . LOCATION TO LEFT OF ARRAY!!!  
A1.2 . A1 = WORD LOCATION  
A2.34 . A2 = QWD LOCATION WITHIN WORD  
A1.1.X11 . CONVERT WORD LOCATION TO ADDRESS  
EXTRACT.A2 . EXTRACT BYTE FROM PROPER QTR  
A3.\*0.X11 . STORE RIGHT-ALIGNED & ZERO-FILLED  
4.X11 .

RETURN

EXTRACT

LA.Q1 A3.0.A1  
LA.Q2 A3.0.A1  
LA.Q3 A3.0.A1  
LA.Q4 A3.0.A1

END

. SUBROUTINE GETCHR( 3 GET ONE CHARACTER FROM CHARACTER STRING  
. 0 KHAR. 3 UNPACKED CHARACTER (LEFT-ALIGNED & SPACE-FILLED)  
. .  
. 1 JSTRNO. 3 CHARACTER STRING  
. ( KHRLOC) 3 CHARACTER LOCATION WITHIN STRING (COUNTING FROM 1 AT LEFT)  
. -----  
. .  
. .  
. ENTRY GETICE( 3 GET INTEGER CHARACTER EQUIVALENT FROM CHARACTER STRING  
. 0 NICE. 3 INTEGER CHARACTER EQUIVALENT (RIGHT-ALIGNED, ZERO-FILLED)  
. .  
. 1 JSTRNO. 3 CHARACTER STRING  
. ( KHRLOC) 3 CHARACTER LOCATION WITHIN STRING (COUNTING FROM 1 AT LEFT)  
. -----  
. .  
. HISTORY  
. -----  
. .  
. E H SCHLOSSER LEC 06/05/78 ORIGINAL CODE  
. .  
. METHOD  
. -----  
. .  
. STRING-RELATIVE CHARACTER LOCATION IS CONVERTED TO MACHINE-DEPENDENT  
. LOCATION AS FOLLOWS:  
. WORD NUMBER = (CHARACTER LOCATION - 1) / CHARACTERS PER WORD  
. CHARACTER LOCATION WITHIN WORD = REMAINDER FROM DIVISION  
. .  
. MACHINE-DEPENDENT CODE  
. -----  
. .  
. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 8-BIT  
. FIELDATA CHARACTERS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
. BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.  
. DIFFERENT COMPILERS (EG., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.  
. .  
. EXTERNAL REFERENCES  
. -----  
. .  
. NONE  
. .  
. EXCEPTIONS  
. -----  
. .  
. 1. KHAR (OR NICE) IS UNCHANGED IF KHRLOC IS LESS THAN 1.  
. .  
. 2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.  
. .  
. .



DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

GETCHR  
002

. GLOBAL DECLARATIONS

. -----

. AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS

. -----

S(01) . 0-BANK . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY  
SPACES

. PROCEDURE

. -----

S(00) . 1-BANK

GETCHR*	LA	A2.*2.X11	. CHAR LOCATION. COUNTING FROM 1
	AA.XU	A2.-1	. CHAR LOCATION. COUNTING FROM 0
	SZ	A1	. DIVISION COMING
	JN	A2.RETCHR	. LOCATION TO LEFT OF STRING!!!
	DI.XU	A1.6	. A1-WORD LOC. A2-SIXTH LOC
	AA	A1.1.X11	. CONVERT WORD LOCATION TO ADDRESS
	EX	EXTRACT.A2	. EXTRACT CHAR FROM PROPER SIXTH
	LA	A2.SPACES	. FILL RESULT ...
	SA	A2.*0.X11	. ... WITH SPACES
	SA.S1	A3.*0.X11	. STORE LEFT-ALIGNED
RETCHR	J	4.X11	.

GETICE*	LA	A2.*2.X11	. CHAR LOCATION. COUNTING FROM 1
	AA.XU	A2.-1	. CHAR LOCATION. COUNTING FROM 0
	SZ	A1	. DIVISION COMING
	JN	A2.RETICE	. LOCATION TO LEFT OF STRING!!!
	DI.XU	A1.6	. A1-WORD LOC. A2-SIXTH LOC
	AA	A1.1.X11	. CONVERT WORD LOCATION TO ADDRESS
	EX	EXTRACT.A2	. EXTRACT CHAR FROM PROPER SIXTH
	SA	A3.*0.X11	. STORE RIGHT-ALIGNED & ZERO-FILLED
RETICE	J	4.X11	.

EXTRACT	LA.S1	A3.0.A1	.
	LA.S2	A3.0.A1	.
	LA.S3	A3.0.A1	.
	LA.S4	A3.0.A1	.
	LA.S5	A3.0.A1	.
	LA.S6	A3.0.A1	.

END

SUBROUTINE GETDBY: 8 GET NON-NEG INTEGER FROM DOUBLE BYTE IN BYTE STRING  
0 IBYT. 8 UNPACKED DOUBLE BYTE (RIGHT-ALIGNED & ZERO-FILLED)  
1 IBYSTR. 8 INTERNAL BYTE STRING  
( IBYLOC) 8 BYTE LOCATION WITHIN STRING OF FIRST BYTE IN DOUBLE BYTE  
8 (COUNTING FROM 1 AT LEFT OF BYTE STRING)  
-----

HISTORY  
-----

E H SCHLOSSER LEC 07/09/79 ORIGINAL CODE

METHOD  
-----

CONVERT STRING-RELATIVE BYTE LOCATION TO MACHINE-DEPENDENT  
LOCATION AS FOLLOWS:  
WORD NUMBER = 34 MOST SIGNIFICANT BITS  
QUARTER-WORD LOCATION WITHIN WORD = 2 LEAST SIGNIFICANT BITS  
EXTRACT 8-BIT CONTENTS OF TWO 9-BIT QUARTER WORDS & CATENATE

MACHINE-DEPENDENT CODE  
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 9-BIT  
QUARTER WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES.  
DIFFERENT COMPILERS (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

EXTERNAL REFERENCES  
-----

NONE

EXCEPTIONS  
-----

1. THE COMPUTER MUST BE IN QUARTER-WORD MODE. (ASSEMBLE WITH 8ASH.F  
OR COLLECT WITH 8MAP.F OR INSURE THAT SETQWD WAS CALLED PREVIOUSLY.)
2. IBYT IS UNCHANGED IF IBYLOC IS LESS THAN 1.
3. THE RESULT IS UNDEFINED IF IBYSTR DOES NOT CONTAIN VALID BYTE VALUES  
(IN THE RANGE BETWEEN 0 AND +255).
4. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

GLOBAL DECLARATIONS

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**GETBYT  
002**

-----

**AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS**

**LOCAL DECLARATIONS**

-----

**NONE**

**PROCEDURE**

-----

<b>S(00) . 1-BANK</b>		<b>. BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY</b>
<b>GETBYT:</b>	<b>LA</b>	<b>A2.*2.X11 . BYTE LOCATION. COUNTING FROM 1</b>
	<b>AA.XU</b>	<b>A2.-1 . BYTE LOCATION. COUNTING FROM 0</b>
	<b>JN</b>	<b>A2.RETURN . LOCATION TO LEFT OF ARRAY!!!</b>
	<b>DSL</b>	<b>A2.2 . A2 = WORD LOCATION</b>
	<b>SSL</b>	<b>A3.3* . A3 = QWD LOCATION WITHIN WORD</b>
	<b>AA</b>	<b>A2.1.X11 . CONVERT WORD LOCATION TO ADDRESS</b>
	<b>EX</b>	<b>GETBYT1.A3 . GET FIRST BYTE OF DOUBLE BYTE</b>
	<b>LSSL</b>	<b>A0.8 . SHIFT 8 BITS LEFT</b>
	<b>EX</b>	<b>ORBYT2.A3 . OR WITH SECOND BYTE</b>
	<b>SA</b>	<b>A1.*0.X11 . STORE RIGHT-ALIGNED &amp; ZERO-FILLED</b>
<b>RETURN</b>	<b>J</b>	<b>4.X11 .</b>
<b>GETBYT1</b>	<b>LA.Q1</b>	<b>A0.0.A2 .</b>
	<b>LA.Q2</b>	<b>A0.0.A2 .</b>
	<b>LA.Q3</b>	<b>A0.0.A2 .</b>
	<b>LA.Q4</b>	<b>A0.0.A2 .</b>
<b>ORBYT2</b>	<b>OR.Q2</b>	<b>A0.0.A2 .</b>
	<b>OR.Q3</b>	<b>A0.0.A2 .</b>
	<b>OR.Q4</b>	<b>A0.0.A2 .</b>
	<b>OR.Q1</b>	<b>A0.1.A2 .</b>
	<b>END</b>	

```

SUBROUTINE GETHEX( 8 GET HEXADECIMAL CHAR FROM NYBLE IN BYTE STRING
0 NYBHEX. 8 HEX DIGIT EQUIVALENT OF NYBLE VALUE
.
1 IBYSTR. 8 BYTE STRING
( IBYLOC. 8 LOCATION OF SUBSTRING WITHIN STRING
( NYBLOC) 8 NYBLE LOCATION OF NYBLE WITHIN SUBSTRING
(BOTH COUNTING FROM 1 AT LEFT OF SUBSTRING)
C -----
C
C
C
C HISTORY
C -----
C
C J C CRISP LEC 07/25/79 REQUIREMENTS
C J C CRISP LEC 08/23/79 ALGORITHM DESIGN
C J C CRISP LEC 08/24/79 ALGORITHM CODING
C
C
C METHOD
C -----
C
C CONVERT NYBLE TO HEXADECIMAL EQUIVALENT AS FOLLOWS:
C CALL GETNYB FOR INTEGER VALUE OF NYBLE
C PERFORM TABLE LOOK-UP FOR HEX EQUIVALENT OF MIN0(NYBLE,15)
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C NONE
C
C
C EXTERNAL REFERENCES
C -----
C
C GETNYB 8 GET NON-NEGATIVE INTEGER FROM NYBLE IN BYTE STRING
C
C
C EXCEPTIONS
C -----
C
C 1. NYBHEX IS UNCHANGED IF IBYLOC AND NYBLOC REFER TO A NYBLE TO THE
C LEFT OF THE BYTE STRING.
C
C 2. THE RESULT IS UNDEFINED IF IBYSTR DOES NOT CONTAIN VALID BYTE
C VALUES (IN THE RANGE BETWEEN 0 AND +255).
C
C
C GLOBAL DECLARATIONS
C -----
C
C INCLUDE MAXINT.LIST 8 MAXIMUM INTEGER VALUE
C
C
C LOCAL DECLARATIONS
C -----

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

GETHEX  
002

```

C      INTEGER IMEX (16)/'0'..'1'..'2'..'3'..'4'..'5'..'6'..'7'..'8'..'9'..
      'A'..'B'..'C'..'D'..'E'..'F'/      & HEX TABLE
      INTEGER NYBLE      & INTEGER VALUE OF NYBLE

C
C
C  PROCEDURE
C  -----
C
C  INITIALIZE NYBLE
C
      NYBLE=MAXINT

C
C
C  GET INTEGER VALUE OF NYBLE
C
      CALL GETNYB (NYBLE, 1BYSTR,(1BYLOC),(NYBLOC))
      IF (NYBLE.EQ.MAXINT) GO TO 900      & NYBLE TO LEFT OF STRING

C
C
C  USE SMALLER OF NYBLE AND 15
C
      NYBLE=MIN0(NYBLE,15)

C
C
C  LOOK UP HEX DIGIT IN TABLE
C
      NYBHEX=IMEX(NYBLE+1)

C
C
900 RETURN
      END

```

DETINT  
Q81

## HISTORY

## METHOD

## • MACHINE-DEPENDENT CODE

## EXTERNAL REFERENCES

## EXCEPTIONS

- ## GLOBAL DECLARATIONS

## LOCAL DECLARATIONS

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

GETINT  
002

```

.      NONE
.
.
.  PROCEDURE
.  -----
.
S(00) . 1-BANK      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
GETINT*      LA      A2,*2,X11      . INTEGER LOCATION. COUNTING FROM 1
              AA,XU      A2,-1      . INTEGER LOCATION. COUNTING FROM 0
              JN      A2,RETURN      . LOCATION TO LEFT OF STRING!!!
              AA      A2,1,X11      . CONVERT LOCATION TO ADDRESS
              LA      A0,0,A2      . A0 := INTSTR(INTLOC)
              SA      A0,*0,X11      . INTEGER := A0
RETURN      J      4,X11      .
              END

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

GETNUL  
001

```

SUBROUTINE GETNUL( 8 DUMMY TRANSFORM TO MATCH GETBYT/GETCHR/GETICE/GETINT
8 8.8.8) 8 3 FORMAL ARGUMENTS TO MATCH GETBYT/GETCHR/GETICE/GETINT
-----
C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEC      12/31/79      ORIGINAL CODE
C
C
C METHOD
C -----
C
C      *** THIS ROUTINE SHOULD NEVER BE CALLED ***
C      GENERATE FATAL ERROR AND INITIATE ERROR WALKBACK.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      UNIVAC FORTRAN V RETURN 0.
C
C
C EXTERNAL REFERENCES
C -----
C
C      MOFATL      8 PRINT/LOG/COUNT 'FATAL ERROR' DIAGNOSTIC MESSAGE
C
C
C EXCEPTIONS
C -----
C
C      NONE.
C
C
C PROCEDURE
C -----
C
C      CALL MOFATL( 'CALL TO GETNUL')
C      RETURN 0
C      END

```



. SUBROUTINE GETNYB: 8 GET NON-NEG INTEGER FROM NYBLE IN BYTE STRING  
. 0 NYBLE. 8 UNPACKED NYBLE (RIGHT-ALIGNED & ZERO-FILLED)  
. .  
. 1 IBYSTR. 8 INTERNAL BYTE STRING  
. 1 IBYLOC. 8 BYTE LOCATION OF SUBSTRING WITHIN STRING  
. 1 NYBLOC) 8 NYBLE LOCATION OF NYBLE WITHIN SUBSTRING  
. 8 (BOTH COUNTING FROM 1 AT LEFT OF STRING/SUBSTRING)  
. -----

. HISTORY  
. -----

. E M SCHLOSSER LEC 07/10/79 ORIGINAL CODE  
. .

. METHOD  
. -----

. CONVERT STRING-RELATIVE NYBLE LOCATION TO MACHINE-DEPENDENT  
. LOCATION AS FOLLOWS:  
. STRING-RELATIVE LOCATION IN NYBLES = 2\*IBYLOC+NYBLOC  
. WORD NUMBER = 33 MOST SIGNIFICANT BITS  
. QUARTER-WORD LOCATION WITHIN WORD = NEXT 2 BITS  
. NYBLE LOCATION WITHIN WORD = NEXT 1 BIT  
. EXTRACT QUARTER WORD & SHIFT OR MASK TO GET PROPER NYBLE

. MACHINE-DEPENDENT CODE  
. -----

. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 9-BIT  
. QUARTER WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
. BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES.  
. DIFFERENT COMPILERS (EO., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

. EXTERNAL REFERENCES  
. -----

. NONE  
. .

. EXCEPTIONS  
. -----

- . 1. THE COMPUTER MUST BE IN QUARTER-WORD MODE. (ASSEMBLE WITH BASH.F  
. OR COLLECT WITH SHAP.F OR INSURE THAT SETQWD WAS CALLED PREVIOUSLY.)
- . 2. NYBLE IS UNCHANGED IF IBYLOC & NYBLOC REFER TO A NYBLE TO THE  
. LEFT OF THE BYTE STRING.
- . 3. THE RESULT IS UNDEFINED IF IBYSTR DOES NOT CONTAIN VALID BYTE VALUES  
. (IN THE RANGE BETWEEN 0 AND +255).

4. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

GLOBAL DECLARATIONS

AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

LOCAL DECLARATIONS

NONE

PROCEDURE

```

S(00) . 1-BANK
OETNYB*   LA      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
          LSSL    A1.*2.X11 . BYTE LOC OF SUBSTRING, COUNTING FROM 1
          AA      A1.1     . NYBLE LOC OF SUBSTRING, COUNTING FROM 2
          AA.XU   A1.*3.X11 . NYBLE LOC OF NYBLE, COUNTING FROM 3
          JN      A1.-3     . NYBLE LOC OF NYBLE, COUNTING FROM 0
          DSL     A1.RETURN . LOCATION TO LEFT OF STRING!!!
          DSL     A1.3      . A1 := WORD LOCATION WITHIN STRING
          SSL     A2.34     . A2 := BYTE LOCATION WITHIN WORD
          AA      A3.35     . A3 := NYBLE LOCATION WITHIN BYTE
          EX      A1.1.X11 . A1 := WORD ADDRESS
          EX      OETBYTE.A2 . A0 := BYTE FROM WORD
          EX      OETNYBLE.A3 . A1 := NYBLE FROM BYTE
          SA      A1.*0.X11 . STORE RIGHT-ALIGNED & ZERO-FILLED
          J        S.X11    .

RETURN
OETBYTE   LA.Q1    A0.0.A1 .
          LA.Q2    A0.0.A1 .
          LA.Q3    A0.0.A1 .
          LA.Q4    A0.0.A1 .

OETNYBLE  DSL      A0.40   . LEFT NYBLE OF BYTE
          AND.U    A0.15   . RIGHT NYBLE OF BYTE

END

```

**BAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**GETQBY  
001**

SUBROUTINE GETQBY: 0 GET INTEGER FROM QUADRUPLE BYTE IN BYTE STRING  
0 IQBYT, 0 UNPACKED QUADRUPLE BYTE (RIGHT-ALIGNED & ZERO-FILLED)  
1 IQBYT, 0 INTERNAL BYTE STRING  
1 IQBYT, 0 BYTE LOCATION WITHIN STRING OF FIRST BYTE IN QUADRUPLE BYTE  
0 (COUNTING FROM 1 AT LEFT OF BYTE STRING)  
-----

**HISTORY**  
-----

E M SCHLOSSER	LEC	07/09/78	ORIGINAL CODE
E M SCHLOSSER	LEC	07/27/78	CLARIFY SIGN OF IQBYT

**METHOD**  
-----

CONVERT STRING-RELATIVE BYTE LOCATION TO MACHINE-DEPENDENT  
LOCATION AS FOLLOWS:  
WORD NUMBER = 34 MOST SIGNIFICANT BITS  
QUARTER-WORD LOCATION WITHIN WORD = 2 LEAST SIGNIFICANT BITS  
EXTRACT 8-BIT CONTENTS OF FOUR 8-BIT QUARTER WORDS & CATENATE

**MACHINE-DEPENDENT CODE**  
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 8-BIT  
QUARTER WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES,  
DIFFERENT COMPILERS (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

**EXTERNAL REFERENCES**  
-----

NONE

**EXCEPTIONS**  
-----

1. THE COMPUTER MUST BE IN QUARTER-WORD MODE. (ASSEMBLE WITH BASH.F  
OR COLLECT WITH SHAP.F OR INSURE THAT SETQWD WAS CALLED PREVIOUSLY.)
2. IQBYT IS UNCHANGED IF IQBYT IS LESS THAN 1.
3. THE SIGN OF IQBYT IS MACHINE-DEPENDENT AND THEREFORE CONSIDERED  
UNDEFINED.
4. THE RESULT IS UNDEFINED IF IQBYT DOES NOT CONTAIN VALID BYTE VALUES  
(IN THE RANGE BETWEEN 0 AND +255).

5. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

GLOBAL DECLARATIONS  
-----

AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

LOCAL DECLARATIONS  
-----

NONE

PROCEDURE  
-----

S(00) . 1-BANK		. BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY	
GETQBY	LA	A2.*2.X11	. BYTE LOCATION. COUNTING FROM 1
	AA.XU	A2.-1	. BYTE LOCATION. COUNTING FROM 0
	JN	A2.RETURN	. LOCATION TO LEFT OF ARRAY!!!
	DSL	A2.2	. A2 = WORD LOCATION
	SSL	A3.34	. A3 = QWD LOCATION WITHIN WORD
	AA	A2.1.X11	. CONVERT WORD LOCATION TO ADDRESS
	EX	GETBYT1.A3	. GET FIRST BYTE OF QUADRUPLE BYTE
	LSSL	A0.0	. SHIFT 0 BITS LEFT
	EX	ORBYT2.A3	. OR WITH SECOND BYTE
	LOSL	A0.44	. SHIFT 1 WORD + 0 BITS LEFT
	EX	ORBYT3.A3	. OR WITH THIRD BYTE
	LOSL	A0.44	. SHIFT 1 WORD + 8 BITS LEFT
	EX	ORBYT4.A3	. OR WITH FOURTH BYTE
	SA	A1.*0.X11	. STORE RIGHT-ALIGNED & ZERO-FILLED
RETURN	J	4.X11	.
. GETBYT1			
	LA.Q1	A0.0.A2	.
	LA.Q2	A0.0.A2	.
	LA.Q3	A0.0.A2	.
	LA.Q4	A0.0.A2	.
. ORBYT2			
	OR.Q2	A0.0.A2	.
. ORBYT3			
	OR.Q3	A0.0.A2	.
. ORBYT4			
	OR.Q4	A0.0.A2	.
	OR.Q1	A0.1.A2	.
	OR.Q2	A0.1.A2	.
	OR.Q3	A0.1.A2	.
. END			

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

GETOKH  
001

```

SUBROUTINE GETOKH( 3 GET NEXT CHAR STRING DATA FIELD FROM IMAGE BUFFER
0 KHFLD. 3 CHAR STRING DATA FIELD (STRIP LEADING SPACES & PAD TO WD BGY)
1 KMLEN. 3 FIXED LENGTH IN CHAR OF KHFLD (EXCL. PAD TO WORD BOUNDARY)
0 LOCLN. 3 POINTERS: (1) LOCTOK: NEXT FIELD LOCATION
                (2) LENTOK: NEXT FIELD LENGTH
                (3) LENIMO: IMAGE BUFFER LENGTH ( *(-1) IF END )
C
C
1 LOCLEP. 3 POINTERS: (1) PREVIOUS FIELD LOCATION
                (2) PREVIOUS FIELD LENGTH
                (3) PREVIOUS IMAGE BUFFER LENGTH ( *(-1) IF END )
C
C
1 IMAGE) 3 IMAGE BUFFER CONTAINING CHARACTER STRING FIELDS SEPARATED BY
THE FOLLOWING DELIMITERS:
C
C      . ENDS DATA FIELD (NEXT DATA FIELD FOLLOWS)
C
C      .. ENDS LAST DATA FIELD IN BLOCK (COMMENT FIELD FOLLOWS)
C
C      ... ENDS LAST FIELD IN BLOCK (NEXT BLOCK FOLLOWS)
C
C -----
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      03/02/75      ORIGINAL CODE IN FIELDS
C      E M SCHLOSSER      LEC      01/17/79      REWRITE & GENERALIZE
C
C METHOD
C -----
C
C THE LOCLN IMAGE POINTERS MUST BE INITIALIZED AS FOLLOWS BEFORE THE FIRST
C CALL TO GETOKH WITH A NEW IMAGE BUFFER:
C      LOCLEP(1)=1
C      LOCLEP(2)=0
C      LOCLEP(3)= (LENGTH OF IMAGE BUFFER IN CHARACTERS)
C
C INPUT ARGUMENTS --
C
C      LOCLEP(1)      LOCLEP(2)      LOCLEP(3)
C BEGIN IMAGE:      1              0      +(LEN IMAGE)
C BEGIN BLOCK:      (LOC BLOCK)    0      +(LEN IMAGE)
C BEGIN DATA FIELD: (LOC NEW FIELD) 0      +(LEN IMAGE)
C CONTINUE SCANNING: (LOC PREV FIELD) (LEN PREV FIELD) +(LEN IMAGE)
C
C ARGUMENTS RETURNED --
C
C      LOCLN(1)      LOCLN(2)      LOCLN(3)      KHFLD
C END IMAGE:      1              0      -(LEN IMAGE)      UNCHANGED
C END BLOCK:      (LOC NEXT BLOCK) 0      -(LEN IMAGE)      UNCHANGED
C DATA FIELD:      (LOC DATA FIELD) (LEN FIELD)      +(LEN IMAGE)      DATA FIELD
C
C GETOKH MAY BE USED TO SCAN ANY NUMBER OF IMAGE BUFFERS CONCURRENTLY.
C PROVIDED EACH HAS ITS OWN LOCLEP/LOCLN POINTER ARRAY(S).
C
C TO ELIMINATE THE NEED FOR EXPLICIT UPDATING OF LOCLEP. THE SAME
C FORMAL ARGUMENT MAY BE USED FOR BOTH LOCLEP AND LOCLN.
C
C

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

OETOKH  
002

```

C MACHINE-DEPENDENT CODE
C -----
C
C     NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C     NEXTOK      & GET POINTERS TO NEXT TOKEN IN IMAGE
C     MOVCSY      & MOVE CHARACTER STRING
C     INTEGER LENPAD  & LENGTH IN CHARACTERS INCLUDING PAD TO WORD BOUNDARY
C
C
C EXCEPTIONS
C -----
C
C     1. IF LOCLN(3) IS LESS THAN +1 ON ENTRY. THEN ALL ARGUMENTS ARE
C        UNCHANGED.
C
C     2. DELIMITER TOKENS WITH MORE THAN 3 CONSECUTIVE COMMAS ARE TREATED AS
C        IF COMPOSED OF 3 COMMAS.
C
C
C GLOBAL DECLARATIONS
C -----
C
C     NONE.
C
C
C LOCAL DECLARATIONS
C -----
C
C     INTEGER LOCLEP(3)      & ARGUMENT
C     INTEGER LOCLN(3)      & ARGUMENT
C     DEFINE LOCTOK=LOCLN(1) & PREVIOUS/CURRENT FIELD LOCATION
C     DEFINE LENTOK=LOCLN(2) & PREVIOUS/CURRENT FIELD LENGTH
C     DEFINE LENIMO=LOCLN(3) & IMAGE BUFFER LENGTH ( +1-1) IF END BLOCK/IMAGE
C     INTEGER IMAGE(1)      & ARGUMENT
C
C
C PROCEDURE
C -----
C
C
C INITIALIZE OUTPUT LOCLN POINTERS
C
C     LOCLN(1)=LOCLEP(1)
C     LOCLN(2)=LOCLEP(2)
C     LOCLN(3)=LOCLEP(3)
C
C
C CHECK FOR END IMAGE/BLOCK CONDITION
C
C     IF(LENIMO.LE.0) GO TO 900
C

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

OCTOKH  
003

```

C
C FIND NEXT TOKEN. CHECK ITS TYPE & LENGTH
C
  100 CALL NEXTOK(KHARI,LOCLEN, LOCLEN,IMAGE, '...', ' ')
      IF(LENIMG.LE.0) GO TO 900      & END OF IMAGE
      IF(KHARI.NE.' ') GO TO 200      & NON-DELIMITER TOKEN
      IF(LENTOK.EQ.1) GO TO 410      & .
      IF(LENTOK.EQ.2) GO TO 420      & ..
                                GO TO 430      & ...

C
C DATA FIELD -- MOVE IT TO KHFLD
C
  200 CALL MOVCST(KHFLD,(1),(LENPAD(KHLEN)),
      - IMAGE,(LOCTOK),(MIN0(LENTOK,KHLEN)), ' ')
      GO TO 900

C
C SINGLE DELIMITER (.) -- GET FOLLOWING DATA FIELD
C
  410 GO TO 100

C
C DOUBLE DELIMITER (..) -- FLUSH FOLLOWING COMMENT FIELD
C
  420 CALL NEXTOK(KHARI,LOCLEN, LOCLEN,IMAGE, '...', ' ')
      IF(LENIMG.LE.0) GO TO 900      & END OF IMAGE
      IF(KHARI.NE.' ') GO TO 420      & COMMENT FIELD
      IF(LENTOK.LT.3) GO TO 420      & NOT END OF BLOCK

C
C TRIPLE DELIMITER (...) -- END OF BLOCK
C
  430 LOCTOK=LOCTOK+LENTOK      & START OF NEXT BLOCK
      LENTOK=0
      LENIMG=-1ABS(LENIMG)

C
C DONE
C
  900 RETURN
      END

```

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**ORABSA  
001**

**(NOT IMPLEMENTED)**



DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

OTDSTS  
001

. SUBROUTINE OTDSTS( 8 GET ARRAY OF NON-NEG INTEGERS FROM BYTE STRING  
. 0 IBYRAY, 8 ARRAY OF UNPACKED BYTES (RIGHT-ALIGNED & ZERO-FILLED)  
. 1 IBYSTR, 8 INTERNAL BYTE STRING  
. 1 IBYLOC, 8 BYTE LOCATION WITHIN STRING (COUNTING FROM 1 AT LEFT)  
. 1 NBYTES) 8 NUMBER OF BYTES  
.....

. HISTORY  
-----

. E H SCHLOSSER LEC 07/07/78 ORIGINAL REQUIREMENTS

. METHOD  
-----

. STRING-RELATIVE BYTE LOCATION IS CONVERTED TO MACHINE-DEPENDENT  
. LOCATION AS FOLLOWS:  
. WORD NUMBER = 34 MOST SIGNIFICANT BITS  
. QUARTER-WORD LOCATION WITHIN WORD = 2 LEAST SIGNIFICANT BITS

. MACHINE-DEPENDENT CODE  
-----

. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 9-BIT  
. QUARTER WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
. BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES,  
. DIFFERENT COMPILERS (EO.. UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

. EXTERNAL REFERENCES  
-----

. NONE

. EXCEPTIONS  
-----

- . 1. THE COMPUTER MUST BE IN QUARTER-WORD MODE. (SEE SUBROUTINES  
SETQWD AND CLRQWD)
- . 2. IBYRAY IS UNCHANGED IF IBYLOC IS LESS THAN 1.
- . 3. IBYRAY IS UNCHANGED IF NBYTES IS LESS THAN 1.
- . 4. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

. GLOBAL DECLARATIONS  
-----

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**070YTS  
002**

**. NONE.**

**.  
.**

**AXRS . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRIES  
S(00) . 1-BANK**

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

ICE  
001

. INTEGER FUNCTION  
0 8 INTEGER-CHARACTER-EQUIVALENT  
- ICEI  
1 KHAR) 8 UNPACKED CHARACTER (LEFT-ALIGNED. REST IGNORED)  
-----

. HISTORY  
-----

. E H SCHLOSSER LEC 06/27/78 ORIGINAL CODE

. METHOD  
-----

. LEFT-ALIGNED CHARACTER IS LOADED INTO REGISTER.

. MACHINE-DEPENDENT CODE  
-----

. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 8-BIT  
. FIELDATA CHARACTERS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
. BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.  
. DIFFERENT COMPILERS (EG.. UNIVAC ASCII FORTRAN). AND DIFFERENT MACHINES.

. EXTERNAL REFERENCES  
-----

. NONE

. EXCEPTIONS  
-----

. 1. IF KHAR IS A CHARACTER STRING INSTEAD OF A SINGLE CHARACTER. THEN THE  
. I-C-E OF THE FIRST CHARACTER IN THE STRING IS RETURNED.

. GLOBAL DECLARATIONS  
-----

. AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS  
-----

. NONE

. PROCEDURE  
-----

**QAM PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**ICE  
002**

**.  
S(00) . 1-BANK . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY  
ICE\* LA.SI AO.\*0.X11 .  
J 2.X11 .  
END**

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

ICHR  
001

. INTEGER FUNCTION  
. 0 8 CHARACTER  
. = ICHR:  
. 1 NICE) 8 INTEGER-CHAR-EQUIV (RIGHT-ALIGNED. REST IGNORED)  
-----  
.   
. HISTORY  
-----  
. E H SCHLOSSER LEC 08/27/78 ORIGINAL CODE  
.   
. METHOD  
-----  
. ICE IS LOADED. RIGHT-ALIGNED & FOLLOWED BY SPACES. THEN  
. SHIFTED LEFT.  
.   
. MACHINE-DEPENDENT CODE  
-----  
. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 8-BIT  
. FIELDATA CHARACTERS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
. BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.  
. DIFFERENT COMPILERS (EO.. UNIVAC ASCII FORTRAN). AND DIFFERENT MACHINES.  
.   
. EXTERNAL REFERENCES  
-----  
. NONE  
.   
. EXCEPTIONS  
-----  
. NONE  
.   
. GLOBAL DECLARATIONS  
-----  
. AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS  
.   
. LOCAL DECLARATIONS  
-----  
. S(01) . D-BANK . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY  
. SPACES . .  
.   
. PROCEDURE

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**ICHR  
002**

. -----

S(00) . 1-BANK	. BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
ICHR*	LA A0.*0.X11 .
	LA A1.SPACES .
	LOSL A0.30 .
	J 2.X11 .
	END

14KONE  
001

## HISTORY

## METHOD

### MACHINE-DEPENDENT CODE

## EXTERNAL REFERENCES

## EXCEPTIONS

## GLOBAL DECLARATIONS

## LOCAL DECLARATIONS

**8-00**

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**14KONE  
002**

```

.
.
. PROCEDURE
. -----
.
S(00) . 1-BANK
14KONE. LA      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
          LMA    A0.0.X11 . A0 := KONE
          JP     A1.01.X11 . A1 := -LENDIT
          TLE.XU A1.WALKBACK . IF LENDIT < 1 THEN WALKBACK
          J      A1.-30     . IF LENDIT > 30
          WALKBACK          . THEN WALKBACK
          LSSL  A0.30.A1    . SHIFT SIGN INTO LEFTMOST BIT AND ...
          SSA   A0.30.A1    . ... THEN SHIFT BACK. EXTENDING SIGN!
RETURN    TZ     3.X11     . SKIP N1 IF RETURN ADDR HAS INVALID INSTR
          J      3.X11     . RETURN
.
WALKBACK  LR      R3.NAMFUN . NAME OF THIS FUNCTION
          SLJ     NERRS     . INITIATE FORTRAN V ERROR WALKBACK
          *       2         . NUMBER OF ARGUMENTS
.
NAMFUN    '14KONE' .
          END

```



14KTWO  
001

## HISTORY

## METHOD

• MACHINE-DEPENDENT CODE  
 #####

**EXTERNAL REFERENCES**  
 \*\*\*\*\*

**EXCEPTIONS**  
**XXXXXXXXXX**

**GLOBAL DECLARATIONS**  
 #####

## LOCAL DECLARATIONS

DAR PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

14KTWO  
002

. NONE.

. PROCEDURE

. -----

S(00) . 1-BANK

14KTWO

LA  
LHA  
JP  
TLE.XU  
J  
LSSL  
SSA  
JP  
ANA.U  
RETURN  
TZ  
J

. BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY

AO.\*0.X11 . AO := KTWO  
A1.\*1.X11 . A1 := -LENDIT  
A1.NALKBCK . IF LENDIT < 1 THEN NALKBCK  
A1.-30 . IF LENDIT > 30  
NALKBCK . THEN NALKBCK  
AO.30.A1 . SHIFT SIGN INTO LEFTMOST BIT AND ...  
AO.30.A1 . ... THEN SHIFT BACK, EXTENDING SIGN:  
AO.RETURN . IF POSITIVE, THEN WE'RE DONE  
AO.1 . IF NEGATIVE, THEN SUBTRACT 1  
3.X11 . SKIP N1 IF RETURN ADDR HAS INVALID INSTR  
3.X11 . RETURN

NALKBCK

LR  
SLJ  
\*

R3.NANFUN  
NERRS  
2

. NAME OF THIS FUNCTION  
. INITIATE FORTRAN V ERROR NALKBCK  
. NUMBER OF ARGUMENTS

NANFUN

\*14KTWO\*  
END

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

KHR4IN  
001

```
      FUNCTION KHR4IN( 8 CHARACTER STRING FOR INTEGER ARGUMENT  
      I IN)          8 INTEGER ARGUMENT  
C -----  
C  
C (E H SCHLOSSER)  
C  
      IMPLICIT INTEGER (A-Z)  
      DEFINE DIGIT=REMAIN/DIV  
C  
      REMAIN=ABS(IN)  
      KHR4IN=0  
      IF(REMAIN.GT.99999) GO TO 900  
      KHR4IN=' 00000'  
      DIV=10000  
      SHIFT=2**24  
100  KHR4IN=KHR4IN+SHIFT*DIGIT  
      REMAIN=REMAIN-DIGIT*DIV  
      DIV=DIV/10  
      SHIFT=SHIFT/2**6  
      IF(SHIFT.NE.0) GO TO 100  
      IF(IN.LT.0) KHR4IN=KHR4IN+'-00000'  
900  RETURN  
      END
```

## **HISTORY**

## METHOD

### MACHINE-DEPENDENT CODE

## EXTERNAL REFERENCES

## EXCEPTIONS

## GLOBAL DECLARATIONS

## LOCAL DECLARATIONS

**NONE.**

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

KONE41  
002

```

.
.
. PROCEDURE
. -----
S(00) . 1-BANK      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
KONE41*             LA      A0,*1.X11      . A0 := INTOER
                   LNA     A1,*0.X11      . A1 := -LENDIT
                   JP      A1,WALKBACK    . IF LENDIT < 1 THEN WALKBACK
                   TLE,XU   A1,-36        . IF LENDIT > 36
                   J        WALKBACK      .          THEN WALKBACK
                   LSSL     A0,36,A1      . SHIFT BACK & FORTH TO ...
                   SSL      A0,36,A1      . ... REMOVE SIGN EXTENSION
RETURN            TZ      3.X11          . SKIP N1 IF RETURN ADDR HAS INVALID INSTR
                   J        3.X11        . RETURN

WALKBACK          LR      R3,NAHFUN      . NAME OF THIS FUNCTION
                   SLJ      NERRS        . INITIATE FORTRAN V ERROR WALKBACK
                   *        2            . NUMBER OF ARGUMENTS

NAHFUN            *KONE41* .
                   END

```

. INTEGER FUNCTION  
. 0 3 RIGHT-ALIGNED TWO'S COMPLEMENT FROM SIGNED INTEGER  
. = KTH041  
. ( LENBIT, 3 LENGTH OF TWO'S-COMPLEMENT NUMBER IN BITS  
. 1 INTEGER) 3 INTEGER  
. -----  
. .

. HISTORY  
. -----

. E H SCHLOSSER LEC 11/19/79 RQMTS/DESIGN/CODE/TEST  
. .

. METHOD  
. -----

. IF NEGATIVE, ADD 1. REMOVE SIGN EXTENSION.  
. .

. MACHINE-DEPENDENT CODE  
. -----

. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 36-BIT  
. WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
. BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.  
. DIFFERENT COMPILERS (EQ., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.  
. .

. EXTERNAL REFERENCES  
. -----

. NERRS 3 UNIVAC FORTRAN V LIBRARY WALKBACK ROUTINE  
. .

. EXCEPTIONS  
. -----

. THE FOLLOWING CONDITIONS INITIATE AN ERROR WALKBACK AND ABORT THE  
. CURRENTLY EXECUTING PROGRAM:

. LENBIT < 0  
. LENBIT > 36  
. MORE THAN 2 ACTUAL ARGUMENTS  
. .

. GLOBAL DECLARATIONS  
. -----

. AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS  
. .

. LOCAL DECLARATIONS  
. -----

. NONE.  
. .

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

KTH041  
002

```

.
.
. PROCEDURE
. -----
.
S(00) . I-BANK      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
KTH041*
    LA              A0.*1.X11      . A0 := INTOER
    LNA             A1.*0.X11      . A1 := -LENBIT
    JP              A1.WALKBACK     . IF LENBIT < 1 THEN WALKBACK
    TLE.XU          A1.-36          . IF LENBIT > 36
    J               WALKBACK        .           THEN WALKBACK
    JP              A0.RETURN        . IF POSITIVE, THEN WE'RE DONE
    JNZ             A0.NEGATIVE      . IF MINUS ZERO, THEN ...
    SZ              A0               . ... CHANGE IT TO PLUS ZERO ...
    J               RETURN          . ... AND WE'RE DONE

.
NEGATIVE           AA.XU            A0.1      . IF NEGATIVE, THEN ADD 1 & ...
                  LSSL             A0.36.A1   . ... SHIFT BACK & FORTH TO ...
                  SSL              A0.36.A1   . ... REMOVE SIGN EXTENSION

.
RETURN            TZ               3.X11      . SKIP N1 IF RETURN ADDR HAS INVALID INSTR
                  J               3.X11      . RETURN

.
WALKBACK          LR              R3.NAMFUN    . NAME OF THIS FUNCTION
                  SLJ             NERRS      . INITIATE FORTRAN V ERROR WALKBACK
                  +               2          . NUMBER OF ARGUMENTS

.
NAMFUN            *KTH041* .
                  END

```

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**LAPD SA  
001**

**(NOT IMPLEMENTED)**



DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

LBYTEQ  
001

INTEGER FUNCTION

0            : LOCATION OF BYTE IN SUBSTRING EQUAL TO SEARCH BYTE  
= LBYTEQ(  
1 JSTRO.     : PACKED BYTE STRING  
( KLOC.     : BYTE LOCATION WITHIN STRING WHERE SUBSTRING BEGINS  
( LEND.     : LENGTH & DIRECTION OF SUBSTRING (POSITIVE = L TO R)  
1 IBYSEA)    : SEARCH BYTE  
              (BYTE LOCATION COUNTED FROM 1 AT LEFT OF STRING)

-----

C HISTORY

-----

C	E H SCHLOSSER	LEC	08/03/79	REQUIREMENTS
C	E H SCHLOSSER	LEC	08/08/79	DESIGN/CODE/TEST

C METHOD

-----

C       COMPARE IBYSEA WITH SUBSTRING. ONE BYTE AT A TIME.

C MACHINE-DEPENDENT CODE

-----

C       NONE.

C EXTERNAL REFERENCES

-----

C       GETBYT       GET BYTE FROM STRING

C EXCEPTIONS

-----

- C       1. ANY PART OF THE SUBSTRING TO THE LEFT OF THE STRING IS IGNORED.
- C       2. LBYTEQ IS SET TO ZERO IF NO EQUAL BYTE IS FOUND.
- C       3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

C GLOBAL DECLARATIONS

-----

C       NONE.

C LOCAL DECLARATIONS

-----

INTEGER INC

: SIGNED INCREMENTATION (+1 OR -1) FOR LBYTEQ

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

LBYTEQ  
002

```

      INTEGER LOCEND      8 BYTE LOCATION OF LAST BYTE AT END OF SUBSTRING
      INTEGER IBYTRY      8 BYTE BEING COMPARED WITH IBYSEA

C
C
C PROCEDURE
C -----
C
C
C INITIALIZE SCAN DIRECTIONS & POINTERS
C
      IF(LEND.EQ.0) GO TO 800
      INC=ISIGN(1,LEND)
      LOCEND=KLOC+LEND-INC
      IBYTRY=-1

C
C
C SCAN & COMPARE BYTES
C
      DO 500 LBYTEQ=KLOC,LOCEND,INC
          CALL GETBYT(IBYTRY, JSTRO,(LBYTEQ))
          IF(IBYTRY.EQ.IBYSEA) GO TO 900
      500 CONTINUE

C
C
C EQUAL BYTE NOT FOUND
C
      800 LBYTEQ=0

C
C
C DONE
C
      900 RETURN
      END

```

LEYTNE  
001

```

C          (BYTE LOCATION COUNTED FROM 1 AT LEFT OF STRING)
C          ~~~~~

```

**E** .....  
.....

**6**

**6**      **\*\*\*\*\***

**6**

**f** .....

**3**

**E** .....

**3**

**E** .....  
.....

.....

1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 2679, 26

.....

[illegible]

DAN PACKAGE APPENDIX R  
 CHAR/BYTE/STRING ROUTINES

LBYTNE  
 002

```

      INTEGER LOCEND      8 BYTE LOCATION OF LAST BYTE AT END OF SUBSTRING
      INTEGER IBYTRY      8 BYTE BEING COMPARED WITH IBYSEA

C
C
C PROCEDURE
C -----
C
C
C INITIALIZE SCAN DIRECTIONS & POINTERS
C
      IF (LEND.EQ.0) GO TO 800
      INC=1SIGN(1,LEND)
      LOCEND=KLOC+LEND-INC
      IBYTRY=IBYSEA

C
C
C SCAN & COMPARE BYTES
C
      DO 500 LBYTNE=KLOC,LOCEND,INC
        CALL GETBYT(IBYTRY, JSTRO,(LBYTNE))
        IF (IBYTRY.NE.IBYSEA) GO TO 900
      500 CONTINUE

C
C
C UNEQUAL BYTE NOT FOUND
C
      800 LBYTNE=0

C
C
C DONE
C
      900 RETURN
      END
  
```

DAH PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

LCHREQ  
001

INTEGER FUNCTION

0            0 LOCATION OF CHAR IN SUBSTRING EQUAL TO SEARCH CHAR  
= LCHREQ(  
1 KSTR0.    0 CHARACTER STRING  
1 KLOC.     0 CHAR LOCATION WITHIN STRING WHERE SUBSTRING BEGINS  
1 LEND.     0 LENGTH & DIRECTION OF SUBSTRING (POSITIVE = L TO R)  
1 KHRSEA)   0 SEARCH CHARACTER  
            (CHAR LOCATION COUNTED FROM 1 AT LEFT OF STRING)

-----

C  
C  
C  
C HISTORY  
C -----

C       E M SCHLOSSER       LEC       08/27/78       ORIGINAL CODE  
C  
C

C  
C METHOD  
C -----

C       COMPARE KHRSEA WITH SUBSTRING. ONE CHARACTER AT A TIME.  
C

C  
C MACHINE-DEPENDENT CODE  
C -----

C       NONE.  
C

C  
C EXTERNAL REFERENCES  
C -----

C       OETCHR       GET CHAR FROM STRING  
C

C  
C EXCEPTIONS  
C -----

- C       1. ANY PART OF THE SUBSTRING TO THE LEFT OF THE STRING IS IGNORED.
- C       2. LCHREQ IS SET TO ZERO IF NO EQUAL CHARACTER IS FOUND.
- C       3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

C  
C GLOBAL DECLARATIONS  
C -----

C       NONE.  
C

C  
C LOCAL DECLARATIONS  
C -----

C  
C       INTEGER INC                   0 SIGNED INCREMENTATION (+1 OR -1) FOR LCHREQ  
C       INTEGER LOCEND               0 CHAR LOCATION OF LAST CHAR AT END OF SUBSTRING

```

      INTEGER KHRTRY      8 CHARACTER BEING COMPARED WITH KHRSEA

C
C
C  PROCEDURE
C  -----
C
C
C  INITIALIZE SCAN DIRECTIONS & POINTERS
C
      IF (LEND.EQ.0) GO TO 800
      INC=ISIGN(1,LEND)
      LOCEND=KLOC+LEND-INC
      KHRTRY='NE'
C
C
C  SCAN & COMPARE CHARACTERS
C
      DO 500 LCHREQ=KLOC,LOCEND,INC
          CALL GETCHR(KHRTRY, KSTR0,(LCHREQ))
          IF (KHRTRY.EQ.KHRSEA) GO TO 900
      500 CONTINUE
C
C
C  EQUAL CHARACTER NOT FOUND
C
      800 LCHREQ=0
C
C
C  000E
C
      900 RETURN
      END
  
```

**LEHANE**  
**001**

**R-104**

BAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

LCHRNE  
002

```
      INTEGER KMRTRY          8 CHARACTER BEING COMPARED WITH KMRSEA
C
C
C PROCEDURE
C -----
C
C
C INITIALIZE SCAN DIRECTIONS & POINTERS
C
      IF(LEND.EQ.0) GO TO 900
      INC=1SIGN(1,LEND)
      LOCEND=KLOC+LEND-INC
      KMRTRY=KMRSEA
C
C
C SCAN & COMPARE CHARACTERS
C
      DO 900 LCHRNE=KLOC,LOCEND,INC
          CALL GETCHR(KMRTRY, KSTR0,(LCHRNE))
          IF(KMRTRY.NE.KMRSEA) GO TO 900
      900 CONTINUE
C
C
C UNEQUAL CHARACTER NOT FOUND
C
      900 LCHRNE=0
C
C
C DONE
C
      900 RETURN
      END
```



**LCSTEQ**  
**001**

```

0      1 LOCATION OF CHAR SUBSTRING 2 IN CHAR SUBSTRING 1
= LCSTEQ1
1 KSTR01. 1 CHARACTER STRING 1
( KLOC1. 1 CHAR LOCATION WITHIN STRING 1 WHERE SUBSTRING 1 BEGINS
( LEND1. 1 LENGTH & DIRECTION OF SUBSTRING 1 (POSITIVE = L TO R)
1 KSTR02. 2 CHARACTER STRING 2
( KLOC2. 2 CHAR LOCATION WITHIN STRING 2 WHERE SUBSTRING 2 BEGINS
( LEND2) 2 LENGTH & DIRECTION OF SUBSTRING 2 (POSITIVE = L TO R)
(CHAR LOCATION COUNTED FROM 1 AT LEFT OF STRING)

```

**R-106**

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

LCSTEQ  
002

```

C -----
C
      INTEGER KP1,KP2      8 CHAR LOCATIONS OF
      INTEGER INC1,INC2    8 SIGNED INCREMENTATION (+1 OR -1) FOR KP1,KP2
      INTEGER NCH1,NCH2    8 NUMBER OF CHARACTERS IN SUBSTRINGS
      INTEGER KVAR2        8 FIRST CHARACTER OF SUBSTRING 2
C
C
C PROCEDURE
C -----
C
C
C INITIALIZE SCAN DIRECTIONS & POINTERS
C
      KP1=KLOC1
      INC1=ISIGN(1,LEND1)
C
      CALL MDEFATL( '***** LCSTEQ NOT YET IMPLEMENTED *****')
      RETURN
      END

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

LENCST  
001

INTEGER FUNCTION

0            % LENGTH IN CHARACTERS OF CHARACTER STRING

% LENCST(

1 NAMEST.   % NAME OF CHARACTER STRING

1 MAXLEN)   % MAXIMUM LENGTH

-----

HISTORY

-----

E H SCHLOSSER	LEC	01/17/79	ORIGINAL CODE WITH NULCST
E H SCHLOSSER	LEC	09/26/79	END STRING WITH NULCST OR NULCHR

METHOD

-----

EXCLUDE NULCST, NULCHR AND TRAILING BLANKS, IF PRESENT, FROM LENGTH.

MACHINE-DEPENDENT CODE

-----

NONE.

EXTERNAL REFERENCES

-----

INTEGER NC4NI	% NUMBER OF CHARACTERS FOR NUMBER OF INTEGERS
INTEGER NI4NC	% NUMBER OF INTEGERS FOR NUMBER OF CHARACTERS
INTEGER LCHREQ	% LOCATE CHARACTER IN SUBSTRING EQUAL TO SEARCH CHAR
INTEGER LCHRNE	% LOCATE CHARACTER IN SUBSTRING NOT EQUAL TO SEARCH CHAR

EXCEPTIONS

-----

1. IF MAXLEN IS LESS THAN 1, THEN A LENGTH OF ZERO IS RETURNED.
2. IF THE STRING IS COMPOSED ONLY OF BLANK(S), THEN A LENGTH OF 1 IS RETURNED.

GLOBAL DECLARATIONS

-----

INCLUDE NULCST.LIST	% DEFINE NULL CHARACTER STRING
INCLUDE NULCHR.LIST	% DEFINE NULL CHARACTER

LOCAL DECLARATIONS

-----

INTEGER NAMEST(1)       % ARGUMENT

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

LENCST  
002

```

      INTEGER LCHRV      & LOCATION OF LAST VALID (NON-NULL) CHARACTER SCANNED
      INTEGER MAXNIN     & MAXIMUM NUMBER OF INTEGERS TO SCAN

C
C
C PROCEDURE
C -----
C
C
C INITIALIZE
C
      LENCST=0
      IF(MAXLEN.LE.0) GO TO 900
C
C
C FIND FIRST NULCST (IF ANY)
C
      LCHRV=0
      MAXNIN=N14NC(MAXLEN)
      DO 300 NINT=1,MAXNIN
          IF(NAMCST(NINT).EQ.NULCST) GO TO 400
          LCHRV=LCHRV+NC4NI(1)
      300 CONTINUE
C
C
C FIND FIRST NULCHR (IF ANY)
C
      400 LCHRV=MIN0(LCHRV,MAXLEN)
      LENCST=LCHREQ(NAMCST,1,LCHRV,NULCHR)-1
      IF(LENCST.LT.0) LENCST=LCHRV
      IF(LENCST.EQ.0) GO TO 900
C
C
C SUBTRACT TRAILING BLANKS FROM LENGTH
C
      LENCST=LCHRNE(NAMCST,LENCST,-LENCST,' ')
      IF(LENCST.EQ.0) LENCST=1      & IF FIRST CHAR IS SPACE, ITS SIGNIFICANT
C
C
C DONE
C
      900 RETURN
      END

```

LENPAD  
001

.....

```
LENPAD=((LENGTH+5)/6)*6      6 6 CHARACTERS PER WORD
RETURN
```

**DAM PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**LENPAD  
002**

**END**

INTEGER FUNCTION

0            8 LOCATION OF ICE IN SUBSTRING EQUAL TO SEARCH ICE  
= LICEEQ(  
1 KSTRO.    8 PACKED CHARACTER STRING  
( KLOC.    8 CHAR LOCATION WITHIN STRING WHERE SUBSTRING BEGINS  
( LEND.    8 LENGTH & DIRECTION OF SUBSTRING (POSITIVE = L TO R)  
1 ICESEA) 8 SEARCH INTEGER-CHARACTER-EQUIVALENT  
            (CHAR LOCATION COUNTED FROM 1 AT LEFT OF STRING)

HISTORY

E H SCHLOSSER    LEC    08/27/78    ORIGINAL CODE

METHOD

COMPARE ICESEA WITH SUBSTRING. ONE CHARACTER AT A TIME.

MACHINE-DEPENDENT CODE

NONE.

EXTERNAL REFERENCES

0ETICE    0ET INTEGER-CHARACTER-EQUIVALENT FROM STRING

EXCEPTIONS

1. ANY PART OF THE SUBSTRING TO THE LEFT OF THE STRING IS IGNORED.
2. LICEEQ IS SET TO ZERO IF NO EQUAL I-C-E IS FOUND.
3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

GLOBAL DECLARATIONS

NONE.

LOCAL DECLARATIONS

INTEGER INC  
INTEGER LOCEND

8 SIGNED INCREMENTATION (+1 OR -1) FOR LICEEQ  
8 CHAR LOCATION OF LAST CHAR AT END OF SUBSTRING

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

LICEEQ  
002

```

      INTEGER ICETRY      8 INTEGER-CHAR-EQUIV BEING COMPARED WITH ICESEA
C
C
C  PROCEDURE
C  -----
C
C
C  INITIALIZE SCAN DIRECTIONS & POINTERS
C
      IF(LEND.EQ.0) GO TO 800
      INC=1SIGN(1,LEND)
      LOCEND=KLOC+LEND-INC
      ICETRY=-1
C
C
C  SCAN & COMPARE INTEGER-CHARACTER-EQUIVALENTS
C
      DO 500 LICEEQ=KLOC,LOCEND,INC
          CALL GETICE(ICETRY, KSTRG,(LICEEQ))
          IF(ICETRY.EQ.ICESEA) GO TO 900
      500 CONTINUE
C
C
C  EQUAL CHARACTER NOT FOUND
C
      800 LICEEQ=0
C
C
C  DONE
C
      900 RETURN
      END

```



```

      INTEGER FUNCTION
      0      8 LOCATION OF ICE IN SUBSTRING NOT EQ TO SEARCH ICE
      * LICENE(
      I KSTRO. 8 PACKED CHARACTER STRING
      I KLOC.  8 CHAR LOCATION WITHIN STRING WHERE SUBSTRING BEGINS
      I LEND.  8 LENGTH & DIRECTION OF SUBSTRING (POSITIVE = L TO R)
      I ICESEA) 8 SEARCH INTEGER-CHARACTER-EQUIVALENT
                  (CHAR LOCATION COUNTED FROM 1 AT LEFT OF STRING)
      -----
C
C
C
C HISTORY
C -----
C      E H SCHLOSSER      LEC      08/27/78      ORIGINAL CODE
C
C
C METHOD
C -----
C      ICESEA IS COMPARED WITH SUBSTRING. ONE CHARACTER AT A TIME.
C
C
C MACHINE-DEPENDENT CODE
C -----
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C      GETICE      8 GET INTEGER-CHARACTER-EQUIVALENT FROM STRING
C
C
C EXCEPTIONS
C -----
C      1. ANY PART OF THE SUBSTRING TO THE LEFT OF THE STRING IS IGNORED.
C      2. LICENSE IS SET TO ZERO IF NO UNEQUAL I-C-E IS FOUND.
C      3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
C
C
C GLOBAL DECLARATIONS
C -----
C      NONE.
C
C
C LOCAL DECLARATIONS
C -----
C
      INTEGER INC      8 SIGNED INCREMENTATION (+1 OR -1) FOR LICENSE
      INTEGER LOCEND    8 CHAR LOCATION OF LAST CHAR AT END OF SUBSTRING

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

LICENE  
002

```

      INTEGER ICETRY      3 INTEGER-CHAR-EQUIV BEING COMPARED WITH ICESEA
C
C
C PROCEDURE
C -----
C
C
C INITIALIZE SCAN DIRECTIONS & POINTERS
C
      IF(LEND.EQ.0) GO TO 800
      INC=1SIGN(1,LEND)
      LOCEND=KLOC+LEND-INC
      ICETRY=ICESEA
C
C
C SCAN & COMPARE INTEGER-CHARACTER-EQUIVALENTS
C
      DO 500 LICENE=KLOC,LOCEND,INC
          CALL GETICE(ICETRY, KSTRG,(LICENE))
          IF(ICETRY.NE.ICESEA) GO TO 900
      500 CONTINUE
C
C
C UNEQUAL CHARACTER NOT FOUND
C
      800 LICENE=0
C
C
C DONE
C
      900 RETURN
      END

```

INTEGER FUNCTION

0            0 LOCATION OF INTEGER IN SUBSTRING EQUAL TO SEARCH INTEGER  
= LINTEQ(  
1 INTSTR.   0 INTEGER STRING  
1 INTLOC.   0 INTEGER LOCATION WITHIN STRING WHERE SUBSTRING BEGINS  
1 LEND.     0 LENGTH & DIRECTION OF SUBSTRING (POSITIVE = L TO R)  
1 INTSEA)   0 SEARCH INTEGER  
             (INTEGER LOCATION COUNTED FROM 1 AT LEFT OF STRING)  
-----

HISTORY  
-----

E M SCHLOSSER	LEC	10/22/79	REQUIREMENTS
E M SCHLOSSER	LEC	10/23/79	DESIGN & CODE

METHOD  
-----

COMPARE INTSEA WITH SUBSTRING. ONE WORD AT A TIME.

MACHINE-DEPENDENT CODE  
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS.  
THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT, BUT  
THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES,  
DIFFERENT COMPILERS (EQ.. UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

EXTERNAL REFERENCES  
-----

NONE.

EXCEPTIONS  
-----

1. ANY PART OF THE SUBSTRING TO THE LEFT OF THE STRING IS IGNORED.
2. LINTEQ IS SET TO ZERO IF NO EQUAL CHARACTER IS FOUND.
3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

GLOBAL DECLARATIONS  
-----

AXRS            . STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS

. -----

. NONE.

. PROCEDURE

. -----

S(00) . 1-BANK			. BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
LINTEQ	LA	A1.*1.X11	. A1 := SUBSTRING LOC. COUNTING FROM 1
	LA	A2.*2.X11	. A2 := SUBSTRING LEND
	LA	A3.*3.X11	. A3 := INTSEA
	JN	A2.MINUS	. CHECK SUBSTRING DIRECTION
PLUS	LX1.U	A0.*1	. A0(INCREMENT) := +1
	J	CHECK	.
MINUS	LX1.U	A0.-1	. A0(INCREMENT) := -1
	LMA	A2.A2	. A2 := ABS(LEND)
	TO	A2.A1	. A2 :=
	LA	A2.A1	. MIN(ABS(LEND).INTLOC)
CHECK	AA.XU	A1.-1	. A1 := SUBSTRING LOC. COUNTING FROM 0
	JN	A1.ERROR	. LOC TO LEFT OF STRING!!!
SEARCH	AA.M2	A1.0.X11	. A1 := ADDRESS OF SUBSTRING
	LR	R1.A2	. R1 := MAX NUMBER OF WORDS TO SEARCH
	LXH	A0.A1	. A0(MODIFIER) := ADDRESS OF SUBSTRING
	SE	A3.0.*A0	. A0(MODIFIER) := ADDR PAST INTSEA IN STR
	J	ERROR	. NOT IN STRING!!!
	TP	A0	. IF SEARCH DIRECTION IS NEGATIVE ...
	AA.U	A0.2	. ... THEN ADDR AFTER := ADDR BEFORE + 2
	LX1.U	A0.0	. A0(INCREMENT) := ZERO
	ANA.M2	A0.0.X11	. A0 := LOC OF INTSEA. COUNTING FROM 1
	J	9.X11	. RETURN LOC OF INTSEA
ERROR	SZ	A0	.
	J	9.X11	. RETURN ZERO
	END		

. INTEGER FUNCTION  
. 0 0 LOCATION OF INTEGER IN SUBSTRING NOT EQUAL TO SEARCH INTEGER  
. 0 LINTNE  
. 1 INTSTR. 0 INTEGER STRING  
. 1 INTLOC. 0 INTEGER LOCATION WITHIN STRING WHERE SUBSTRING BEGINS  
. 1 LENG. 0 LENGTH & DIRECTION OF SUBSTRING (POSITIVE = L TO R)  
. 1 INTSEA) 0 SEARCH INTEGER  
. (INTEGER LOCATION COUNTED FROM 1 AT LEFT OF STRING)  
-----

. HISTORY  
-----

. E M SCHLOSSER LEC 10/22/79 REQUIREMENTS  
. E M SCHLOSSER LEC 10/23/79 DESIGN & CODE

. METHOD  
-----

. COMPARE INTSEA WITH SUBSTRING. ONE WORD AT A TIME.

. MACHINE-DEPENDENT CODE  
-----

. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS.  
. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT, BUT  
. THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES.  
. DIFFERENT COMPILERS (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

. EXTERNAL REFERENCES  
-----

. NONE.

. EXCEPTIONS  
-----

- . 1. ANY PART OF THE SUBSTRING TO THE LEFT OF THE STRING IS IGNORED.
- . 2. LINTNE IS SET TO ZERO IF NO UNEQUAL CHARACTER IS FOUND.
- . 3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

. GLOBAL DECLARATIONS  
-----

. AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS

. -----

. NONE.

. PROCEDURE

. -----

S(00) . 1-BANK			. BANK REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
LINTNE	LA	A1.*1.X11	. A1 := SUBSTRING LOC. COUNTING FROM 1
	LA	A2.*2.X11	. A2 := SUBSTRING LEND
	LA	A3.*3.X11	. A3 := INTSEA
	JN	A2.MINUS	. CHECK SUBSTRING DIRECTION
PLUS	LX1.U	A0.*1	. A0(INCREMENT) := +1
	J	CHECK	.
MINUS	LX1.U	A0.-1	. A0(INCREMENT) := -1
	LMA	A2.A2	. A2 := ABS(LEND)
	TO	A2.A1	. A2 :=
	LA	A2.A1	. MIN(ABS(LEND),INTLOC)
CHECK	AA.XU	A1.-1	. A1 := SUBSTRING LOC. COUNTING FROM 0
	JN	A1.ERROR	. LOC TO LEFT OF STRING!!!
SEARCH	AA.M2	A1.0.X11	. A1 := ADDRESS OF SUBSTRING
	LR	R1.A2	. R1 := MAX NUMBER OF WORDS TO SEARCH
	LXM	A0.A1	. A0(MODIFIER) := ADDRESS OF SUBSTRING
	SNE	A3.0.*A0	. A0(MODIFIER) := ADDR PAST INT NE IN STR
	J	ERROR	. NO UNEQUAL INTEGER IN STRING!!!
	TP	A0	. IF SEARCH DIRECTION IS NEGATIVE ...
	AA.U	A0.2	. ... THEN ADDR AFTER := ADDR BEFORE + 2
	LX1.U	A0.0	. A0(INCREMENT) := ZERO
	ANA.M2	A0.0.X11	. A0 := LOC OF INT NE INTSEA. COUNT FROM 1
	J	S.X11	. RETURN LOC OF INT NE INTSEA
ERROR	SZ	A0	.
	J	S.X11	. RETURN ZERO
	END		

INTEGER FUNCTION

0            3 NUMBER OF SUBSTRING LOWER IN COLLATING SEQUENCE:

C            0    KSTR01 = KSTR02  
C            1    KSTR01 < KSTR02  
C            2    KSTR02 < KSTR01

= LOWCST(

1 KSTR01,   3 CHARACTER STRING 1  
( KPOS1,   3 CHAR POSITION WITHIN STRING 1 WHERE SUBSTRING 1 BEGINS  
( LEND1,   3 LENGTH & DIRECTION OF SUBSTRING 1 (POSITIVE = L TO R)  
1 KSTR02,   3 CHARACTER STRING 2  
( KPOS2,   3 CHAR POSITION WITHIN STRING 2 WHERE SUBSTRING 2 BEGINS  
( LEND2)   3 LENGTH & DIRECTION OF SUBSTRING 2 (POSITIVE = L TO R)  
            (CHAR POSITION COUNTED FROM 1 AT LEFT OF STRING)

C -----  
C  
C  
C

C HISTORY

C -----

C            E H SCHLOSSER    LEC    06/27/78    ORIGINAL CODE  
C  
C

C METHOD

C -----

C            DOWNHILE CHARACTERS REMAINING AND I-C-E DIFFERENCE IS ZERO  
C            GET NEXT I-C-E'S FROM 2 SUBSTRINGS & COMPUTE DIFFERENCE  
C            (IF A SUBSTRING IS EXHAUSTED, TREAT IT AS IF PADDED WITH SPACES)  
C            NORMALIZE I-C-E DIFFERENCE  
C  
C

C MACHINE-DEPENDENT CODE

C -----

C            NONE.  
C  
C

C EXTERNAL REFERENCES

C -----

C            ICE            INTEGER CHARACTER EQUIVALENT FUNCTION FOR CHARACTER  
C            GETICE        GET ICE FROM STRINGS  
C  
C

C EXCEPTIONS

C -----

- C            1. IF THERE IS NO CORRESPONDING CHARACTER IN A SUBSTRING, THE CHARACTER
- C            SPACE IS ASSUMED.
- C
- C            2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
- C
- C

C GLOBAL DECLARATIONS

C -----

C

DAM PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

LOWCST  
002

```

C      NONE.
C
C
C LOCAL DECLARATIONS
C -----
C
      INTEGER KP1,KP2          8 CHAR POSITIONS OF CHARACTERS BEING COMPARED
      INTEGER INC1,INC2        8 SIGNED INCREMENTATION (+1 OR -1) FOR KP1,KP2
      INTEGER NCH1,NCH2        8 NUMBER OF CHARACTERS IN SUBSTRINGS
      INTEGER NCHMAX           8 NUMBER OF CHARACTERS IN LONGEST SUBSTRING
      INTEGER ICEPAD           8 INTEGER-CHAR-EQUIV OF BLANK TO PAD SHORT SUBSTR
      INTEGER NCH              8 NUMBER OF CHARACTERS COMPARED
      INTEGER NICE1,NICE2      8 I-C-E'S OF CHARACTERS BEING COMPARED
C
C
C PROCEDURE
C -----
C
C
C INITIALIZE SCAN DIRECTIONS & POINTERS
C
      KP1=KPOS1
      INC1=ISIGN(1,LEND1)
      NCH1=ABS(LEND1)
      KP2=KPOS2
      INC2=ISIGN(1,LEND2)
      NCH2=ABS(LEND2)
      NCHMAX=MAX0(NCH1,NCH2)
      ICEPAD=ICE(' ')
      LOWCST=0
C
C
C SCAN & COMPARE CHARACTERS
C
      DO 500 NCH=1,NCHMAX
        NICE1=ICEPAD
        IF(NCH.LE.NCH1) CALL GETICE(NICE1, KSTR01,(KP1))
        KP1=KP1+INC1
        NICE2=ICEPAD
        IF(NCH.LE.NCH2) CALL GETICE(NICE2, KSTR02,(KP2))
        KP2=KP2+INC2
        LOWCST=NICE1-NICE2
        IF(LOWCST.NE.0) GO TO 900
      500 CONTINUE
C
C
C NORMALIZE RESULT
C
      900 IF(LOWCST.GT.0) LOWCST=2
        IF(LOWCST.LT.0) LOWCST=1
        RETURN
      END

```



```

SUBROUTINE MOVBST( 3 MOVE BYTES TO OUTPUT SUBSTRING FROM INPUT SUBSTRING
0 JSTOUT. 3 OUTPUT BYTE STRING
1 LOCOUT. 3 BYTE LOCATION IN OUTPUT STRING WHERE OUTPUT SUBSTRING BEGINS
1 LENOUT. 3 LENGTH & DIRECTION OF OUTPUT SUBSTRING (POSITIVE = L TO R)
.
1 JSTIN. 3 INPUT BYTE STRING
1 LOCIN. 3 BYTE LOCATION WITHIN INPUT STRING WHERE INPUT SUBSTRING BEGINS
1 LENIN. 3 LENGTH & DIRECTION OF INPUT SUBSTRING (POSITIVE = L TO R)
1 IBYPAD) 3 BYTE TO PAD OUTPUT SUBSTRING ON EXHAUSTING INPUT SUBSTRING
(BYTE LOCATIONS COUNTED FROM 1 AT LEFT OF STRING)
-----

```

#### HISTORY

```

-----
J C CRISP          LEC      07/25/79    REQUIREMENTS
E H SCHLOSSER      LEC      07/27/79    DESIGN
E H SCHLOSSER      LEC      07/30/79    CODE/TEST

```

#### METHOD

```

-----
CONVERT STRING-RELATIVE BYTE LOCATIONS TO MACHINE-DEPENDENT
LOCATIONS AS FOLLOWS:
    WORD NUMBER = 34 MOST SIGNIFICANT BITS
    QUARTER-WORD LOCATION WITHIN WORD = 2 LEAST SIGNIFICANT BITS
DETERMINE CO-ROUTINES & INITIAL ENTRY POINTS TO GET/PUT BYTES.
MOVE BYTES WITH GET/PUT CO-ROUTINES.
PAD BYTES WITH PAD/PUT CO-ROUTINES.

```

#### MACHINE-DEPENDENT CODE

```

-----
WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 9-BIT
QUARTER WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.
BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES,
DIFFERENT COMPILERS (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

```

#### EXTERNAL REFERENCES

```

-----
NONE.

```

#### EXCEPTIONS

- ```

-----
1. THE COMPUTER MUST BE IN QUARTER-WORD MODE. (ASSEMBLE WITH 8ASH.F
   OR COLLECT WITH 8MAP.F OR INSURE THAT SETQWD WAS CALLED PREVIOUSLY.)
2. IF THERE IS NO CORRESPONDING BYTE IN INPUT SUBSTRING, IBYPAD IS

```

MOVED TO OUTPUT SUBSTRING.

3. THIS TRANSFORM MAY BE USED TO REVERSE THE ORDER OF THE CONTENTS OF A SUBSTRING BY SPECIFYING OPPOSITE DIRECTIONS FOR THE OUTPUT AND INPUT SUBSTRINGS. PROVIDED THESE SUBSTRINGS DO NOT OVERLAP. IF THE SUBSTRINGS DO OVERLAP THE RESULTS OF THE ATTEMPTED REVERSAL WILL BE UNDEFINED.
4. THIS TRANSFORM MAY BE USED TO SHIFT A SUBSTRING WITHIN A STRING BY USING THE SAME ACTUAL ARGUMENT FOR JSTOUT AND JSTIN AND BY SPECIFYING OUTPUT AND INPUT SUBSTRINGS WHICH PARTIALLY OVERLAP. PROVIDED THE DIRECTION OF THE SHIFT IS OPPOSITE TO THE DIRECTION OF THE SUBSTRINGS.
5. WHEN SHIFTING A SUBSTRING TO THE LEFT (LOCOUT<LOCIN) AND PADDING AT THE RIGHT. THEN BOTH LENOUT AND LENIN MUST BE POSITIVE (DIRECTED TO THE RIGHT) OR THE RESULTS WILL BE UNDEFINED.
6. WHEN SHIFTING A SUBSTRING TO THE RIGHT (LOCOUT>LOCIN) AND PADDING AT THE LEFT. THEN BOTH LENOUT AND LENIN MUST BE NEGATIVE (DIRECTED TO THE LEFT) OR THE RESULTS WILL BE UNDEFINED.
7. IF EITHER SUBSTRING LIES WHOLLY OR PARTIALLY OUTSIDE ITS STRING THE RESULTS WILL BE UNDEFINED.
8. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

GLOBAL DECLARATIONS

AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

LOCAL DECLARATIONS

\$(00) . D-BANK  
XISAVE RES 1 . SAVE LOCATION FOR REGISTER XI

PROCEDURE

\$(01) . I-BANK  
NOVBST SX XI.XISAVE . SAVE REGISTER XI

COMPUTE INITIAL WORD ADDRESS & BYTE LOCATION WITHIN WORD FOR INPUT SUBSTRING

|       |           |                                  |
|-------|-----------|----------------------------------|
| LA    | A0.04.X11 | . BYTE LOCIN. COUNTING FROM 1    |
| AA.XU | A0.-1     | . BYTE LOCIN. COUNTING FROM 0    |
| JN    | A0.RETURN | . RETURN IF TO LEFT OF STRING    |
| DSL   | A0.2      | . A0 := WORD LOCIN WITHIN STRING |
| SSL   | A1.34     | . A1 := BYTE LOCIN WITHIN WORD   |
| AA    | A0.3.X11  | . A0 := WORD ADDRESS IN          |

MSI.U      A1.3      . A1 := 3-BYTE LOCIN WITHIN WORD

. COMPUTE INITIAL WORD ADDRESS & BYTE LOCATION WITHIN WORD FOR OUTPUT SUBSTRING

|       |           |                                          |
|-------|-----------|------------------------------------------|
| LA    | A2.*1.X11 | . BYTE LOCOUT. COUNTING FROM 1           |
| AA.XU | A2.-1     | . BYTE LOCOUT. COUNTING FROM 0           |
| JN    | A2.RETURN | . RETURN IF TO LEFT OF STRING            |
| DSL   | A2.2      | . A2 := WORD LOCOUT WITHIN WITHIN STRING |
| SSL   | A3.34     | . A3 := BYTE LOCOUT WITHIN WORD          |
| AA    | A2.0.X11  | . A2 := WORD ADDRESS OUT                 |
| MSI.U | A3.2      | . A3 := 2-BYTE LOCOUT WITHIN WORD        |

. DETERMINE CO-ROUTINE & INITIAL ENTRY POINT TO GET BYTES

|         |        |             |                                          |
|---------|--------|-------------|------------------------------------------|
| LENIND  | LA     | A4.*5.X11   | . LENIND                                 |
|         | JN     | A4.INMINUS  | .                                        |
| INPLUS  | AA     | A1.(RGETQ1) | . A1 := CO-ROUTINE ADDR TO GET NEXT BYTE |
|         | LX1.XU | A0.*1       | . WORD INCREMENTATION IS +1              |
|         | J      | LENOUD      | .                                        |
| INMINUS | LNA    | A1.A1       | .                                        |
|         | AA     | A1.(LGETQ1) | . A1 := CO-ROUTINE ADDR TO GET NEXT BYTE |
|         | LX1.XU | A0.-1       | . WORD INCREMENTATION IS -1              |
|         | LNA    | A4.A4       | . A4 := # OF BYTES IN                    |

. DETERMINE CO-ROUTINE & INITIAL ENTRY POINT TO PUT BYTES

|          |        |             |                                          |
|----------|--------|-------------|------------------------------------------|
| LENOUD   | LA     | A5.*2.X11   | . LENOUD                                 |
|          | JN     | A5.OUTMINUS | .                                        |
| OUTPLUS  | AA     | A3.(RPUTQ1) | . A3 := CO-ROUTINE ADDR TO PUT NEXT BYTE |
|          | LX1.XU | A2.*1       | . WORD INCREMENTATION IS +1              |
|          | J      | MOVNPAD     | .                                        |
| OUTMINUS | LNA    | A3.A3       | .                                        |
|          | AA     | A3.(LPUTQ1) | . A3 := CO-ROUTINE ADDR TO PUT NEXT BYTE |
|          | LX1.XU | A2.-1       | . WORD INCREMENTATION IS -1              |
|          | LNA    | A5.A5       | . A5 := # OF BYTES OUT                   |

. INITIALIZE REPEAT REGISTERS FOR MOVING AND PADDING

|         |       |       |                                  |
|---------|-------|-------|----------------------------------|
| MOVNPAD | TO    | A4.A5 | . SKIP NEXT INSTR IF A5 > A4     |
|         | LA    | A4.A5 | . A4 := # OF BYTES TO MOVE       |
|         | ANA   | A5.A4 | . A5 := # OF BYTES TO PAD        |
|         | ANA.U | A4.1  | . A4 := (A-1) OF BYTES TO MOVE   |
|         | LNA   | A4.A4 | . A4 := -(A-1) OF BYTES TO MOVE  |
|         | LXM   | X1.A4 | . X1M := -(A-1) OF BYTES TO MOVE |
|         | LX1.U | X1.1  | . X11 := +1                      |
|         | J     | 0.A1  | . START MOVING BYTES!!!          |

. CO-ROUTINE TO GET BYTES FROM SUBSTRING DIRECTED TO THE LEFT

|        |      |           |   |
|--------|------|-----------|---|
| LGETQ4 | LNJ  | A1.0.A3   | . |
|        | JM01 | X1.GETPAD | . |

OAM PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

NOV85T/ASH  
004

```

      LR.Q4      R1.0.A0      . GET Q4
      LMJ        A1.0.A3      .
LOETQ3      JH01      X1.0ETPAD .
      LR.Q3      R1.0.A0      . GET Q3
      LMJ        A1.0.A3      .
LOETQ2      JH01      X1.0ETPAD .
      LR.Q2      R1.0.A0      . GET Q2
      LMJ        A1.0.A3      .
LOETQ1      JH01      X1.0ETPAD .
      LR.Q1      R1.0.*A0     . GET Q1 & DECREMENT WORD *
      J          LOETQ4-1     .

```

.  
CO-ROUTINE TO PUT BYTES INTO SUBSTRING DIRECTED TO THE LEFT

```

      LMJ        A3.0.A1      .
LPUTQ4      SR.Q4      R1.0.A2      . PUT Q4
      LMJ        A3.0.A1      .
LPUTQ3      SR.Q3      R1.0.A2      . PUT Q3
      LMJ        A3.0.A1      .
LPUTQ2      SR.Q2      R1.0.A2      . PUT Q2
      LMJ        A3.0.A1      .
LPUTQ1      SR.Q1      R1.0.*A2     . PUT Q1 & DECREMENT WORD *
      J          LPUTQ4-1     .

```

.  
CO-ROUTINE TO GET BYTES FROM SUBSTRING DIRECTED TO THE RIGHT

```

      LMJ        A1.0.A3      .
RGETQ1      JH01      X1.0ETPAD .
      LR.Q1      R1.0.A0      . GET Q1
      LMJ        A1.0.A3      .
RGETQ2      JH01      X1.0ETPAD .
      LR.Q2      R1.0.A0      . GET Q2
      LMJ        A1.0.A3      .
RGETQ3      JH01      X1.0ETPAD .
      LR.Q3      R1.0.A0      . GET Q3
      LMJ        A1.0.A3      .
RGETQ4      JH01      X1.0ETPAD .
      LR.Q4      R1.0.*A0     . GET Q4 & INCREMENT WORD *
      J          RGETQ1-1     .

```

.  
CO-ROUTINE TO PUT BYTES INTO SUBSTRING DIRECTED TO THE RIGHT

```

      LMJ        A3.0.A1      .
RPUTQ1      SR.Q1      R1.0.A2      . PUT Q1
      LMJ        A3.0.A1      .
RPUTQ2      SR.Q2      R1.0.A2      . PUT Q2
      LMJ        A3.0.A1      .
RPUTQ3      SR.Q3      R1.0.A2      . PUT Q3
      LMJ        A3.0.A1      .
RPUTQ4      SR.Q4      R1.0.*A2     . PUT Q4 & INCREMENT WORD *
      J          RPUTQ1-1     .

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

NOV88T/ASH  
005

. CO-ROUTINE TO GET PAD BYTES

|        |     |             |                                    |
|--------|-----|-------------|------------------------------------|
| GETPAD | LR  | R1,0,X11    | . R1 := PAD BYTE                   |
|        | LA  | A1,(PUTPAD) | . A1 := ADDR OF PUTPAD             |
| PUTPAD | JGD | AS,0,A3     | . PUT BYTE & DECREMENT AS TIL ZERO |

. TERMINATION CODE

|        |    |           |              |
|--------|----|-----------|--------------|
| RETURN | LX | X1,X1SAVE | . RESTORE X1 |
|        | J  | 0,X11     | .            |
|        |    | END       |              |

```

SUBROUTINE MOVBST( 8 MOVE BYTES TO OUTPUT SUBSTRING FROM INPUT SUBSTRING
0 JSTOUT, 8 OUTPUT BYTE STRING
( LOCOUT, 8 BYTE LOCATION IN OUTPUT STRING WHERE OUTPUT SUBSTRING BEGINS
( LENOUT, 8 LENGTH & DIRECTION OF OUTPUT SUBSTRING (POSITIVE = L TO R)
=
I JSTIN, 8 INPUT BYTE STRING
( LOCIN, 8 BYTE LOCATION WITHIN INPUT STRING WHERE INPUT SUBSTRING BEGINS
( LENIN, 8 LENGTH & DIRECTION OF INPUT SUBSTRING (POSITIVE = L TO R)
I IBYPAD) 8 BYTE TO PAD OUTPUT SUBSTRING ON EXHAUSTING INPUT SUBSTRING
(BYTE LOCATIONS COUNTED FROM 1 AT LEFT OF STRING)
-----

```

C  
C  
C  
C  
C HISTORY  
C -----

|               |     |          |              |
|---------------|-----|----------|--------------|
| J C CRISP     | LEC | 07/25/79 | REQUIREMENTS |
| E H SCHLOSSER | LEC | 07/27/79 | DESIGN       |
| E H SCHLOSSER | LEC | 07/30/79 | CODE/TEST    |

C  
C  
C  
C  
C METHOD  
C -----

MOVE INPUT SUBSTRING TO OUTPUT SUBSTRING, ONE BYTE AT A TIME.

C  
C  
C  
C  
C MACHINE-DEPENDENT CODE  
C -----

NONE.

C  
C  
C  
C  
C EXTERNAL REFERENCES  
C -----

GETBYT 8 GET BYTE FROM INPUT STRING  
PUTBYT 8 PUT BYTE INTO OUTPUT STRING

C  
C  
C  
C  
C EXCEPTIONS  
C -----

1. IF THERE IS NO CORRESPONDING BYTE IN INPUT SUBSTRING, IBYPAD IS MOVED TO OUTPUT SUBSTRING.
2. THIS TRANSFORM MAY BE USED TO REVERSE THE ORDER OF THE CONTENTS OF A SUBSTRING BY SPECIFYING OPPOSITE DIRECTIONS FOR THE OUTPUT AND INPUT SUBSTRINGS. PROVIDED THESE SUBSTRINGS DO NOT OVERLAP. IF THE SUBSTRINGS DO OVERLAP THE RESULTS OF THE ATTEMPTED REVERSAL WILL BE UNDEFINED.
3. THIS TRANSFORM MAY BE USED TO SHIFT A SUBSTRING WITHIN A STRING BY USING THE SAME ACTUAL ARGUMENT FOR JSTOUT AND JSTIN AND BY SPECIFYING OUTPUT AND INPUT SUBSTRINGS WHICH PARTIALLY OVERLAP. PROVIDED THE DIRECTION OF THE SHIFT IS OPPOSITE TO THE DIRECTION OF THE SUBSTRINGS.

```

C
C 4. WHEN SHIFTING A SUBSTRING TO THE LEFT (LOCOUT<LOCIN) AND PADDING
C    AT THE RIGHT, THEN BOTH LENOUT AND LENIND MUST BE POSITIVE (DIRECTED
C    TO THE RIGHT) OR THE RESULTS WILL BE UNDEFINED.
C
C 5. WHEN SHIFTING A SUBSTRING TO THE RIGHT (LOCOUT>LOCIN) AND PADDING
C    AT THE LEFT, THEN BOTH LENOUT AND LENIND MUST BE NEGATIVE (DIRECTED
C    TO THE LEFT) OR THE RESULTS WILL BE UNDEFINED.
C
C 6. IF EITHER SUBSTRING LIES WHOLLY OR PARTIALLY OUTSIDE ITS STRING THE
C    RESULTS WILL BE UNDEFINED.
C
C 7. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
C
C GLOBAL DECLARATIONS
C -----
C
C    NONE.
C
C LOCAL DECLARATIONS
C -----
C
C    INTEGER LBYOUT,LBYIN      3 BYTE LOCATIONS OF BYTE BEING MOVED
C    INTEGER INCOUT,INCIN      3 SIGNED INCREMENTATION (+1/-1) FOR LBYOUT,LBYIN
C    INTEGER NBYOUT,NBYIN      3 NUMBER OF BYTES IN SUBSTRINGS
C    INTEGER NBY              3 NUMBER OF BYTES MOVED
C    INTEGER IBYT              3 BYTE BEING MOVED
C
C
C C PROCEDURE
C -----
C
C C INITIALIZE SCAN DIRECTIONS & POINTERS
C
C    LBYOUT=LOCOUT
C    INCOUT=1SIGN(1,LENOUT)
C    NBYOUT=1ABS(LENOUT)
C    LBYIN=LOCIN
C    INCIN=1SIGN(1,LENIND)
C    NBYIN=1ABS(LENIND)
C
C
C C MOVE BYTES
C
C    DO 500 NBY=1,NBYOUT
C        IBYT=IBYPAD
C        IF(NBY.LE.NBYIN) CALL GETBYT(IBYT,  JSTIN,LBYIN)
C        LBYIN=LBYIN+INCIN
C        IF(NBY.LE.NBYOUT) CALL PUTBYT(IJSTOUT,LBYOUT,  IBYT)
C        LBYOUT=LBYOUT+INCOUT
C    500 CONTINUE
C
C
C

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

NOV86T/F6R  
003

C DONE  
C  
RETURN  
END



SUBROUTINE MOVBYT( 8 MOVE BYTE TO OUTPUT STRING FROM INPUT STRING  
8 JSTOUT. 8 OUTPUT BYTE STRING  
( LOCOUT. 8 BYTE LOCATION WITHIN OUTPUT STRING  
8  
8 JSTIN. 8 INPUT BYTE STRING  
( LOCIN) 8 BYTE LOCATION WITHIN INPUT STRING  
(BYTE LOCATIONS COUNTED FROM 1 AT LEFT OF STRING)  
-----

HISTORY  
-----

|               |     |          |                  |
|---------------|-----|----------|------------------|
| J C CRISP     | LEC | 07/29/79 | REQUIREMENTS     |
| E H SCHLOSSER | LEC | 07/27/79 | DESIGN/CODE/TEST |

METHOD  
-----

CONVERT STRING-RELATIVE BYTE LOCATION TO MACHINE-DEPENDENT  
LOCATION AS FOLLOWS:  
WORD NUMBER = 34 MOST SIGNIFICANT BITS  
QUARTER-WORD LOCATION WITHIN WORD = 2 LEAST SIGNIFICANT BITS  
LOAD & STORE REQUIRED QUARTER WORD.

MACHINE-DEPENDENT CODE  
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 9-BIT  
QUARTER WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES.  
DIFFERENT COMPILERS (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

EXTERNAL REFERENCES  
-----

NONE

EXCEPTIONS  
-----

1. THE COMPUTER MUST BE IN QUARTER-WORD MODE. (ASSEMBLE WITH 8ASH.F  
OR COLLECT WITH 8MAP.F OR INSURE THAT SETQWD WAS CALLED PREVIOUSLY.)
2. JSTOUT IS UNCHANGED IF LOCOUT OR LOCIN ARE LESS THAN 1.
3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

GLOBAL DECLARATIONS  
-----

NOVBYT  
002

**A-131**

NOVCHR  
001

## HISTORY

## METHOD

## MACHINE-DEPENDENT CODE

## EXTERNAL REFERENCES

## EXCEPTIONS

- ## GLOBAL DECLARATIONS

**R-132**

DAM PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

NOVCHR  
882

. LOCAL DECLARATIONS  
-----

NONE.

. PROCEDURE  
-----

|                |       |                                                   |
|----------------|-------|---------------------------------------------------|
| 8:001 . 1-BANK |       | . BANKS REVERSED TO ELIMINATE INDIRECT LOAD ENTRY |
| NOVCHR*        | LA    | A1.*3.X11 . CHAR LOCIN. COUNTING FROM 1           |
|                | AA.XU | A1.-1 . CHAR LOCIN. COUNTING FROM 0               |
|                | SZ    | A0 . DIVISION COMING                              |
|                | JN    | A1.5.X11 . RETURN IF LOCIN TO LEFT OF STRING      |
|                | 01.U  | A0.6 . A0 := WORD LOCIN WITHIN STRING             |
|                |       | A1 := CHAR LOCIN WITHIN WORD                      |
|                | AA    | A0.2.X11 . A0 := WORD ADDRESS IN                  |
|                | EX    | EXTRACT.A1 . EXTRACT CHAR FROM PROPER SIXTH WORD  |
|                | LA    | A1.*1.X11 . CHAR LOCOUT. COUNTING FROM 1          |
|                | AA.XU | A1.-1 . CHAR LOCOUT. COUNTING FROM 0              |
|                | SZ    | A0 . DIVISION COMING                              |
|                | JN    | A1.5.X11 . RETURN IF LOCOUT TO LEFT OF STRING     |
|                | 01.U  | A0.6 . A0 := WORD LOCOUT WITHIN STRING            |
|                |       | A1 := CHAR LOCOUT WITHIN WORD                     |
|                | AA    | A0.0.X11 . A0 := WORD ADDRESS OUT                 |
|                | EX    | INSERT.A1 . INSERT CHAR INTO PROPER SIXTH WORD    |
|                | J     | 5.X11 . RETURN                                    |
| EXTRACT        | LA.S1 | A3.0.A0 .                                         |
|                | LA.S2 | A3.0.A0 .                                         |
|                | LA.S3 | A3.0.A0 .                                         |
|                | LA.S4 | A3.0.A0 .                                         |
|                | LA.S5 | A3.0.A0 .                                         |
|                | LA.S6 | A3.0.A0 .                                         |
| INSERT         | SA.S1 | A3.0.A0 .                                         |
|                | SA.S2 | A3.0.A0 .                                         |
|                | SA.S3 | A3.0.A0 .                                         |
|                | SA.S4 | A3.0.A0 .                                         |
|                | SA.S5 | A3.0.A0 .                                         |
|                | SA.S6 | A3.0.A0 .                                         |
|                | END   |                                                   |

ORIGINAL PAGE IS  
OF POOR QUALITY

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

MOVCST/ASH  
001

SUBROUTINE MOVCST( 8 MOVE CHARS TO OUTPUT SUBSTRING FROM INPUT SUBSTRING  
0 KSTOUT. 8 OUTPUT CHAR STRING  
( LOCOUT. 8 CHAR LOCATION IN OUTPUT STRING WHERE OUTPUT SUBSTRING BEGINS  
( LENOUT. 8 LENGTH & DIRECTION OF OUTPUT SUBSTRING (POSITIVE = L TO R)  
1 KSTIN. 8 INPUT CHAR STRING  
( LOCIN. 8 CHAR LOCATION WITHIN INPUT STRING WHERE INPUT SUBSTRING BEGINS  
( LENIND. 8 LENGTH & DIRECTION OF INPUT SUBSTRING (POSITIVE = L TO R)  
1 KHRPAD) 8 CHAR TO PAD OUTPUT SUBSTRING ON EXHAUSTING INPUT SUBSTRING  
(CHAR LOCATIONS COUNTED FROM 1 AT LEFT OF STRING)  
-----

HISTORY  
-----

|               |     |          |              |
|---------------|-----|----------|--------------|
| J C CRISP     | LEC | 07/25/79 | REQUIREMENTS |
| E H SCHLOSSER | LEC | 07/27/79 | DESIGN       |
| E H SCHLOSSER | LEC | 07/30/79 | CODE/TEST    |

METHOD  
-----

CONVERT STRING-RELATIVE CHARACTER LOCATIONS TO MACHINE-DEPENDENT  
LOCATIONS AS FOLLOWS:  
WORD NUMBER = (CHARACTER LOCATION - 1) / CHARACTERS PER WORD  
SIXTH-WORD LOCATION WITHIN WORD = REMAINDER FROM DIVISION  
DETERMINE CO-ROUTINES & INITIAL ENTRY POINTS TO GET/PUT CHARACTERS.  
MOVE CHARACTERS WITH GET/PUT CO-ROUTINES.  
PAD CHARACTERS WITH PAD/PUT CO-ROUTINES.

MACHINE-DEPENDENT CODE  
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 8-BIT  
FIELDATA CHARACTERS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES,  
DIFFERENT COMPILERS (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

EXTERNAL REFERENCES  
-----

NONE.

EXCEPTIONS  
-----

1. IF THERE IS NO CORRESPONDING CHARACTER IN INPUT SUBSTRING, KHRPAD IS  
MOVED TO OUTPUT SUBSTRING.
2. THIS TRANSFORM MAY BE USED TO REVERSE THE ORDER OF THE CONTENTS OF A

- . SUBSTRING BY SPECIFYING OPPOSITE DIRECTIONS FOR THE OUTPUT AND INPUT
- . SUBSTRINGS. PROVIDED THESE SUBSTRINGS DO NOT OVERLAP. IF THE
- . SUBSTRINGS DO OVERLAP THE RESULTS OF THE ATTEMPTED REVERSAL WILL BE
- . UNDEFINED.
- . 3. THIS TRANSFORM MAY BE USED TO SHIFT A SUBSTRING WITHIN A STRING BY
- . USING THE SAME ACTUAL ARGUMENT FOR KSTOUT AND KSTIN AND BY SPECIFYING
- . OUTPUT AND INPUT SUBSTRINGS WHICH PARTIALLY OVERLAP. PROVIDED THE
- . DIRECTION OF THE SHIFT IS OPPOSITE TO THE DIRECTION OF THE SUBSTRINGS.
- . 4. WHEN SHIFTING A SUBSTRING TO THE LEFT (LOCOUT<LOCIN) AND PADDING
- . AT THE RIGHT, THEN BOTH LENOUT AND LENIND MUST BE POSITIVE (DIRECTED
- . TO THE RIGHT) OR THE RESULTS WILL BE UNDEFINED.
- . 5. WHEN SHIFTING A SUBSTRING TO THE RIGHT (LOCOUT>LOCIN) AND PADDING
- . AT THE LEFT, THEN BOTH LENOUT AND LENIND MUST BE NEGATIVE (DIRECTED
- . TO THE LEFT) OR THE RESULTS WILL BE UNDEFINED.
- . 6. IF EITHER SUBSTRING LIES WHOLLY OR PARTIALLY OUTSIDE ITS STRING THE
- . RESULTS WILL BE UNDEFINED.
- . 7. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

GLOBAL DECLARATIONS

AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

LOCAL DECLARATIONS

S(00) . D-BANK

XISAVE RES 1 . SAVE LOCATION FOR REGISTER X1

PROCEDURE

S(01) . I-BANK

NOVCST SX X1.XISAVE . SAVE REGISTER X1

COMPUTE INITIAL WORD ADDRESS & CHAR LOCATION WITHIN WORD FOR INPUT SUBSTRING

|       |           |                                  |
|-------|-----------|----------------------------------|
| LA    | A1.*4.X11 | . CHAR LOCIN. COUNTING FROM 1    |
| AA.XU | A1.-1     | . CHAR LOCIN. COUNTING FROM 0    |
| SZ    | A0        | . DIVISION COMING                |
| JN    | A1.RETURN | . RETURN IF TO LEFT OF STRING    |
| DI.U  | A0.8      | . A0 := WORD LOCIN WITHIN STRING |
|       |           | . A1 := CHAR LOCIN WITHIN WORD   |
| AA    | A0.3.X11  | . A0 := WORD ADDRESS IN          |
| NSI.U | A1.3      | . A1 := 3*CHAR LOCIN WITHIN WORD |

. COMPUTE INITIAL WORD ADDRESS & CHAR LOCATION WITHIN WORD FOR OUTPUT SUBSTRING

|       |           |                                          |
|-------|-----------|------------------------------------------|
| LA    | A3.*1.X11 | . CHAR LOCOUT, COUNTING FROM 1           |
| AA.XU | A3.-1     | . CHAR LOCOUT, COUNTING FROM 0           |
| SZ    | A2        | . DIVISION COMING                        |
| JN    | A3.RETURN | . RETURN IF TO LEFT OF STRING            |
| DI.U  | A2.6      | . A2 := WORD LOCOUT WITHIN WITHIN STRING |
|       |           | . A3 := CHAR LOCOUT WITHIN WORD          |
| AA    | A2.0.X11  | . A2 := WORD ADDRESS OUT                 |
| MSI.U | A3.2      | . A3 := 2*CHAR LOCOUT WITHIN WORD        |

. DETERMINE CO-ROUTINE & INITIAL ENTRY POINT TO GET CHARS

|         |        |             |                                          |
|---------|--------|-------------|------------------------------------------|
| LENIND  | LA     | A4.*5.X11   | . LENIND                                 |
|         | JN     | A4.INMINUS  | .                                        |
| INPLUS  | AA     | A1.(ROETS1) | . A1 := CO-ROUTINE ADDR TO GET NEXT CHAR |
|         | LXI.XU | A0.*1       | . WORD INCREMENTATION IS +1              |
|         | J      | LENOUD      | .                                        |
| INMINUS | LNA    | A1.A1       | .                                        |
|         | AA     | A1.(LOETS1) | . A1 := CO-ROUTINE ADDR TO GET NEXT CHAR |
|         | LXI.XU | A0.-1       | . WORD INCREMENTATION IS -1              |
|         | LNA    | A4.A4       | . A4 := # OF CHARS IN                    |

. DETERMINE CO-ROUTINE & INITIAL ENTRY POINT TO PUT CHARS

|          |        |             |                                          |
|----------|--------|-------------|------------------------------------------|
| LENOUD   | LA     | A5.*2.X11   | . LENOUD                                 |
|          | JN     | A5.OUTHINUS | .                                        |
| OUTPLUS  | AA     | A3.(RPUTS1) | . A3 := CO-ROUTINE ADDR TO PUT NEXT CHAR |
|          | LXI.XU | A2.*1       | . WORD INCREMENTATION IS +1              |
|          | J      | MOVNPAD     | .                                        |
| OUTHINUS | LNA    | A3.A3       | .                                        |
|          | AA     | A3.(LPUTS1) | . A3 := CO-ROUTINE ADDR TO PUT NEXT CHAR |
|          | LXI.XU | A2.-1       | . WORD INCREMENTATION IS -1              |
|          | LNA    | A5.A5       | . A5 := # OF CHARS OUT                   |

. INITIALIZE REPEAT REGISTERS FOR MOVING AND PADDING

|         |       |       |                                  |
|---------|-------|-------|----------------------------------|
| MOVNPAD | TO    | A4.A5 | . SKIP NEXT INSTR IF A5 > A4     |
|         | LA    | A4.A5 | . A4 := # OF CHARS TO MOVE       |
|         | ANA   | A5.A4 | . A5 := # OF CHARS TO PAD        |
|         | ANA.U | A4.1  | . A4 := (0-1) OF CHARS TO MOVE   |
|         | LNA   | A4.A4 | . A4 := -(0-1) OF CHARS TO MOVE  |
|         | LXN   | X1.A4 | . X1N := -(0-1) OF CHARS TO MOVE |
|         | LXI.U | X1.1  | . X11 := +1                      |
|         | J     | 0.A1  | . START MOVING CHARS!!!          |

. CO-ROUTINE TO GET CHARS FROM SUBSTRING DIRECTED TO THE LEFT

|        |       |           |          |
|--------|-------|-----------|----------|
| LOETS6 | LMJ   | A1.0.A3   | .        |
|        | JNGI  | X1.0ETPAD | .        |
|        | LM.S6 | R1.0.A0   | . GET S6 |

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

NOVCST/ASH  
884

|        |       |           |                             |
|--------|-------|-----------|-----------------------------|
|        | LMJ   | A1.0.A3   | .                           |
| LOETS5 | JM01  | X1.0ETPAD | .                           |
|        | LR.S5 | R1.0.A0   | . GET S5                    |
|        | LMJ   | A1.0.A3   | .                           |
| LOETS4 | JM01  | X1.0ETPAD | .                           |
|        | LR.S4 | R1.0.A0   | . GET S4                    |
|        | LMJ   | A1.0.A3   | .                           |
| LOETS3 | JM01  | X1.0ETPAD | .                           |
|        | LR.S3 | R1.0.A0   | . GET S3                    |
|        | LMJ   | A1.0.A3   | .                           |
| LOETS2 | JM01  | X1.0ETPAD | .                           |
|        | LR.S2 | R1.0.A0   | . GET S2                    |
|        | LMJ   | A1.0.A3   | .                           |
| LOETS1 | JM01  | X1.0ETPAD | .                           |
|        | LR.S1 | R1.0.*A0  | . GET S1 & DECREMENT WORD * |
|        | J     | LOETS6-1  | .                           |

.  
CO-ROUTINE TO PUT CHARS INTO SUBSTRING DIRECTED TO THE LEFT

|        |       |          |                             |
|--------|-------|----------|-----------------------------|
|        | LMJ   | A3.0.A1  | .                           |
| LPUTS6 | SR.S6 | R1.0.A2  | . PUT S6                    |
|        | LMJ   | A3.0.A1  | .                           |
| LPUTS5 | SR.S5 | R1.0.A2  | . PUT S5                    |
|        | LMJ   | A3.0.A1  | .                           |
| LPUTS4 | SR.S4 | R1.0.A2  | . PUT S4                    |
|        | LMJ   | A3.0.A1  | .                           |
| LPUTS3 | SR.S3 | R1.0.A2  | . PUT S3                    |
|        | LMJ   | A3.0.A1  | .                           |
| LPUTS2 | SR.S2 | R1.0.A2  | . PUT S2                    |
|        | LMJ   | A3.0.A1  | .                           |
| LPUTS1 | SR.S1 | R1.0.*A2 | . PUT S1 & DECREMENT WORD * |
|        | J     | LPUTS6-1 | .                           |

.  
CO-ROUTINE TO GET CHARS FROM SUBSTRING DIRECTED TO THE RIGHT

|        |       |           |                             |
|--------|-------|-----------|-----------------------------|
|        | LMJ   | A1.0.A3   | .                           |
| ROETS1 | JM01  | X1.0ETPAD | .                           |
|        | LR.S1 | R1.0.A0   | . GET S1                    |
|        | LMJ   | A1.0.A3   | .                           |
| ROETS2 | JM01  | X1.0ETPAD | .                           |
|        | LR.S2 | R1.0.A0   | . GET S2                    |
|        | LMJ   | A1.0.A3   | .                           |
| ROETS3 | JM01  | X1.0ETPAD | .                           |
|        | LR.S3 | R1.0.A0   | . GET S3                    |
|        | LMJ   | A1.0.A3   | .                           |
| ROETS4 | JM01  | X1.0ETPAD | .                           |
|        | LR.S4 | R1.0.A0   | . GET S4                    |
|        | LMJ   | A1.0.A3   | .                           |
| ROETS5 | JM01  | X1.0ETPAD | .                           |
|        | LR.S5 | R1.0.A0   | . GET S5                    |
|        | LMJ   | A1.0.A3   | .                           |
| ROETS6 | JM01  | X1.0ETPAD | .                           |
|        | LR.S6 | R1.0.*A0  | . GET S6 & INCREMENT WORD * |
|        | J     | ROETS1-1  | .                           |



DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

NOVCST/ASH  
008

.  
.  
. CO-ROUTINE TO PUT CHARS INTO SUBSTRING DIRECTED TO THE RIGHT  
.

|        |       |          |                             |
|--------|-------|----------|-----------------------------|
|        | LNJ   | A3.0.A1  | .                           |
| RPUTS1 | SR.S1 | R1.0.A2  | . PUT S1                    |
|        | LNJ   | A3.0.A1  | .                           |
| RPUTS2 | SR.S2 | R1.0.A2  | . PUT S2                    |
|        | LNJ   | A3.0.A1  | .                           |
| RPUTS3 | SR.S3 | R1.0.A2  | . PUT S3                    |
|        | LNJ   | A3.0.A1  | .                           |
| RPUTS4 | SR.S4 | R1.0.A2  | . PUT S4                    |
|        | LNJ   | A3.0.A1  | .                           |
| RPUTS5 | SR.S5 | R1.0.A2  | . PUT S5                    |
|        | LNJ   | A3.0.A1  | .                           |
| RPUTS6 | SR.S6 | R1.0.*A2 | . PUT S6 & INCREMENT WORD * |
|        | J     | RPUTS1-1 | .                           |

.  
.  
. CO-ROUTINE TO GET PAD CHARS  
.

|        |       |             |                                    |
|--------|-------|-------------|------------------------------------|
| GETPAD | LR.S1 | R1.*8.X11   | . R1 := PAD CHAR                   |
|        | LA    | A1.(PUTPAD) | . A1 := ADDR OF PUTPAD             |
| PUTPAD | J00   | A5.0.A3     | . PUT CHAR & DECREMENT AS TIL ZERO |

.  
.  
. TERMINATION CODE  
.

|        |    |           |              |
|--------|----|-----------|--------------|
| RETURN | LX | X1.X1SAVE | . RESTORE X1 |
|        | J  | 8.X11     | .            |

.  
END

SUBROUTINE NOVCST( 3 MOVE CHARS TO OUTPUT SUBSTRING FROM INPUT SUBSTRING  
0 KSTOUT, 3 OUTPUT CHARACTER STRING  
( LOCOUT, 3 CHAR LOCATION IN OUTPUT STRING WHERE OUTPUT SUBSTRING BEGINS  
( LENOUT, 3 LENGTH & DIRECTION OF OUTPUT SUBSTRING (POSITIVE = L TO R)  
.  
1 KSTIN, 3 INPUT CHARACTER STRING  
( LOGIN, 3 CHAR LOCATION WITHIN INPUT STRING WHERE INPUT SUBSTRING BEGINS  
( LENIND, 3 LENGTH & DIRECTION OF INPUT SUBSTRING (POSITIVE = L TO R)  
1 KHRPAD) 3 CHAR TO PAD OUTPUT SUBSTRING ON EXHAUSTING INPUT SUBSTRING  
(CHAR LOCATIONS COUNTED FROM 1 AT LEFT OF STRING)  
-----

C  
C  
C  
C  
C HISTORY  
C -----

C E M SCHLOSSER LEC 06/27/78 ORIGINAL CODE  
C E M SCHLOSSER LEC 07/27/79 NOMENCLATURE & EXCEPTIONS FOR SHIFTS  
C  
C

C METHOD  
C -----

C MOVE INPUT SUBSTRING TO OUTPUT SUBSTRING. ONE CHARACTER AT A TIME.  
C

C MACHINE-DEPENDENT CODE  
C -----

C NONE.  
C

C EXTERNAL REFERENCES  
C -----

C GETCHR 3 GET CHARACTER FROM INPUT STRING  
C PUTCHR 3 PUT CHARACTER INTO OUTPUT STRING  
C

C EXCEPTIONS  
C -----

- C 1. IF THERE IS NO CORRESPONDING CHARACTER IN INPUT SUBSTRING. KHRPAD IS  
C MOVED TO OUTPUT SUBSTRING.
- C 2. THIS TRANSFORM MAY BE USED TO REVERSE THE ORDER OF THE CONTENTS OF A  
C SUBSTRING BY SPECIFYING OPPOSITE DIRECTIONS FOR THE OUTPUT AND INPUT  
C SUBSTRINGS. PROVIDED THESE SUBSTRINGS DO NOT OVERLAP. IF THE  
C SUBSTRINGS DO OVERLAP THE RESULTS OF THE ATTEMPTED REVERSAL WILL BE  
C UNDEFINED.
- C 3. THIS TRANSFORM MAY BE USED TO SHIFT A SUBSTRING WITHIN A STRING BY  
C USING THE SAME ACTUAL ARGUMENT FOR KSTOUT AND KSTIN AND BY SPECIFYING  
C OUTPUT AND INPUT SUBSTRINGS WHICH PARTIALLY OVERLAP. PROVIDED THE  
C DIRECTION OF THE SHIFT IS OPPOSITE TO THE DIRECTION OF THE SUBSTRINGS.  
C

```

C      4. WHEN SHIFTING A SUBSTRING TO THE LEFT (LOCOUT<LOCIN) AND PADDING
C      AT THE RIGHT, THEN BOTH LENOUT AND LENIND MUST BE POSITIVE (DIRECTED
C      TO THE RIGHT) OR THE RESULTS WILL BE UNDEFINED.
C
C      5. WHEN SHIFTING A SUBSTRING TO THE RIGHT (LOCOUT>LOCIN) AND PADDING
C      AT THE LEFT, THEN BOTH LENOUT AND LENIND MUST BE NEGATIVE (DIRECTED
C      TO THE LEFT) OR THE RESULTS WILL BE UNDEFINED.
C
C      6. IF EITHER SUBSTRING LIES WHOLLY OR PARTIALLY OUTSIDE ITS STRING THE
C      RESULTS WILL BE UNDEFINED.
C
C      7. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER LCHOUT,LCMIN      1 CHAR LOCATIONS OF CHARACTER BEING MOVED
C      INTEGER INCOUT,INCIN      1 SIGNED INCREMENTATION (+1/-1) FOR LCHOUT,LCMIN
C      INTEGER NCHOUT,NCHIN      1 NUMBER OF CHARACTERS IN SUBSTRINGS
C      INTEGER NCH              1 NUMBER OF CHARACTERS MOVED
C      INTEGER KCHAR            1 CHARACTER BEING MOVED
C
C
C C PROCEDURE
C -----
C
C C INITIALIZE SCAN DIRECTIONS & POINTERS
C
C      LCHOUT=LOCOUT
C      INCOUT=1SIGN(1,LENOUT)
C      NCHOUT=1ABS(LENOUT)
C      LCMIN=LOCIN
C      INCIN=1SIGN(1,LENIND)
C      NCHIN=1ABS(LENIND)
C
C
C C MOVE CHARACTERS
C
C      DO 500 NCH=1,NCHOUT
C          KCHAR=KCHARPAD
C          IF(NCH.LE.NCHIN) CALL GETCHR(KCHAR, KSTIN,LCMIN)
C          LCMIN=LCMIN+INCIN
C          IF(NCH.LE.NCHOUT) CALL PUTCHR(KSTOUT,LCHOUT, KCHAR)
C          LCHOUT=LCHOUT+INCOUT
C      500 CONTINUE
C
C
C C DONE

```

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**NOVCS7/FCR  
003**

**C**

**RETURN  
END**

. SUBROUTINE MOVDBY: 8 MOVE DOUBLE BYTE TO OUTPUT STRING FROM INPUT STRING  
. 0 JSTOUT. 8 OUTPUT BYTE STRING  
. 1 LOCOUT. 8 BYTE LOCATION WITHIN OUTPUT STRING WHERE DOUBLE BYTE BEGINS  
. 1 JSTIN. 8 INPUT BYTE STRING  
. 1 LOCIN) 8 BYTE LOCATION WITHIN INPUT STRING WHERE DOUBLE BYTE BEGINS  
. (BYTE LOCATIONS COUNTED FROM 1 AT LEFT OF STRING)  
-----

. HISTORY  
-----

. J C CRISP LEC 07/25/79 REQUIREMENTS  
. E H SCHLOSSER LEC 07/27/79 DESIGN/CODE/TEST

. METHOD  
-----

. CONVERT STRING-RELATIVE BYTE LOCATION TO MACHINE-DEPENDENT  
. LOCATION AS FOLLOWS:  
. WORD NUMBER = 34 MOST SIGNIFICANT BITS  
. QUARTER-WORD LOCATION WITHIN WORD = 8 LEAST SIGNIFICANT BITS  
. LOAD WORD(S) CONTAINING TWO REQUIRED ADJACENT QUARTER WORDS.  
. SHIFT IF NECESSARY TO GET BOTH INTO RIGHT HALF OF REGISTER.  
. STORE REGISTER INTO ONE HALF WORD IF POSSIBLE. OTHERWISE INTO TWO  
. QUARTER WORDS.

. MACHINE-DEPENDENT CODE  
-----

. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 9-BIT  
. QUARTER WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
. BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES.  
. DIFFERENT COMPILERS (EO.. UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

. EXTERNAL REFERENCES  
-----

. NONE

. EXCEPTIONS  
-----

- . 1. THE COMPUTER MUST BE IN QUARTER-WORD MODE. (ASSEMBLE WITH BASH.F  
. OR COLLECT WITH SNAP.F OR INSURE THAT SETQWD WAS CALLED PREVIOUSLY.)
- . 2. JSTOUT IS UNCHANGED IF LOCOUT OR LOCIN ARE LESS THAN 1.
- . 3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

. GLOBAL DECLARATIONS  
.-----

AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS  
.-----

NONE.

. PROCEDURE  
.-----

|                 |       |            |                                                         |
|-----------------|-------|------------|---------------------------------------------------------|
| \$(00) . 1-BANK |       |            | . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY |
| NOVDBY*         | LA,U  | A1,0       | . (BIT 35 MUST BE 0 FOR DSL/SSL BELOW)                  |
|                 | LA    | A0,03.X11  | . BYTE LOCIN. COUNTING FROM 1                           |
|                 | AA,XU | A0,-1      | . BYTE LOCIN. COUNTING FROM 0                           |
|                 | JN    | A0,5.X11   | . RETURN IF LOCIN TO LEFT OF STRING                     |
|                 | DSL   | A0,2       | . A0 := WORD LOCIN WITHIN STRING                        |
|                 | SSL   | A1,33      | . A1 := 2*BYTE LOCIN WITHIN WORD                        |
|                 | AA    | A0,2.X11   | . A0 := WORD ADDRESS IN                                 |
|                 | J     | GETDB12.A1 | . GET DBYTE FROM PROPER LOC                             |
| GETDB12         | LA,M1 | A2,0.A0    | . GET BYTES 1 & 2 FROM WORD                             |
|                 | J     | PUT        | .                                                       |
| GETDB23         | LA    | A2,0.A0    | . GET FULL WORD AND ...                                 |
|                 | J     | SHIFT10    | . ... SHIFT BYTES 2 & 3 INTO POSITION                   |
| GETDB34         | LA,M2 | A2,0.A0    | . GET BYTES 3 & 4 FROM WORD                             |
|                 | J     | PUT        | .                                                       |
| GETDB45         | OL    | A2,0.A0    | . GET 2 WORDS AND ...                                   |
|                 | LDL   | A2,10      | . ... SHIFT BYTES 4 & 5 ...                             |
| SHIFT10         | SSL   | A2,0       | . ... INTO POSITION                                     |
| PUT             | LA    | A0,01.X11  | . BYTE LOCOUT. COUNTING FROM 1                          |
|                 | AA,XU | A0,-1      | . BYTE LOCOUT. COUNTING FROM 0                          |
|                 | JN    | A0,5.X11   | . RETURN IF LOCOUT TO LEFT OF STRING                    |
|                 | DSL   | A0,2       | . A0 := WORD LOCOUT WITHIN STRING                       |
|                 | SSL   | A1,32      | . A1 := 4*BYTE LOCOUT WITHIN WORD                       |
|                 | AA    | A0,0.X11   | . A0 := WORD ADDRESS OUT                                |
|                 | J     | PUTDB12.A1 | . PUT DBYTE INTO PROPER LOC                             |
| PUTDB12         | SA,M1 | A2,0.A0    | . PUT INTO BYTES 1 & 2 OF WORD                          |
|                 | J     | S,X11      | . RETURN                                                |
|                 | NOP   |            | . SLACK WORD (NEEDED FOR ALIGNMENT)                     |
|                 | NOP   |            | . SLACK WORD (NEEDED FOR ALIGNMENT)                     |
| PUTDB23         | SA,Q3 | A2,0.A0    | . PUT INTO BYTES 3 ...                                  |
|                 | SSL   | A2,0       | . ... AND ...                                           |
|                 | SA,Q2 | A2,0.A0    | . ...2 OF WORD                                          |
|                 | J     | S,X11      | . RETURN                                                |
| PUTDB34         | SA,M2 | A2,0.A0    | . PUT INTO BYTES 3 & 4 OF WORD                          |
|                 | J     | S,X11      | . RETURN                                                |
|                 | NOP   |            | . SLACK WORD (NEEDED FOR ALIGNMENT)                     |
|                 | NOP   |            | . SLACK WORD (NEEDED FOR ALIGNMENT)                     |

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

NOV08Y  
003

PUTDB4S      SA.Q1      A2.1.A0      . PUT INTO BYTE 1 OF NEXT WORD ...  
              SSL        A2.0        . ... AND ...  
              SA.Q4      A2.0.A0      . ... BYTE 4 OF THIS WORD  
              J          S.X11      . RETURN  
  
              END

```

SUBROUTINE MOVIST( 0 MOVE INTEGERS TO OUTPUT SUBSTRING FROM INPUT SUBSTR
0 IOSTUT, 0 OUTPUT INTEGER STRING
1 LOCOUT, 0 INTEGER LOCATION IN OUTPUT STRING WHERE OUTPUT SUBSTRING BEGIN
1 LENOUT, 0 LENGTH & DIRECTION OF OUTPUT SUBSTRING (POSITIVE = L TO R)
1
1 IOSTIN, 0 INPUT INTEGER STRING
1 LOCIN, 0 INTEGER LOCATION WITHIN INPUT STRING WHERE INPUT SUBSTRING BEG
1 LENIN, 0 LENGTH & DIRECTION OF INPUT SUBSTRING (POSITIVE = L TO R)
1 INTPAD) 0 INTEGER TO PAD OUTPUT SUBSTRING ON EXHAUSTING INPUT SUBSTRING
(INTEGER LOCATIONS COUNTED FROM 1 AT LEFT OF STRING)
-----

```

#### HISTORY

-----

|               |     |          |                  |
|---------------|-----|----------|------------------|
| E M SCHLOSSER | LEC | 08/03/70 | REQUIREMENTS     |
| E M SCHLOSSER | LEC | 08/30/70 | DESIGN/CODE/TEST |

#### METHOD

-----

THIS TRANSFORM USES THE UNIVAC 1100 SERIES BLOCK TRANSFER INSTRUCTION.  
IT IS FASTER THAN A FORTRAN 'DO' LOOP FOR MOVES INVOLVING MORE THAN  
10 WORDS.

#### MACHINE-DEPENDENT CODE

-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING ON:  
30-BIT WORD PER INTEGER. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-  
INDEPENDENT BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC  
FORTRAN V. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES.  
DIFFERENT COMPILERS (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

#### EXTERNAL REFERENCES

-----

NONE.

#### EXCEPTIONS

-----

1. IF THERE IS NO CORRESPONDING INTEGER IN INPUT SUBSTRING, INTPAD IS  
MOVED TO OUTPUT SUBSTRING.
2. THIS TRANSFORM MAY BE USED TO REVERSE THE ORDER OF THE CONTENTS OF A  
SUBSTRING BY SPECIFYING OPPOSITE DIRECTIONS FOR THE OUTPUT AND INPUT  
SUBSTRINGS. PROVIDED THESE SUBSTRINGS DO NOT OVERLAP. IF THE  
SUBSTRINGS DO OVERLAP THE RESULTS OF THE ATTEMPTED REVERSAL WILL BE  
UNDEFINED.

ORIGINAL PAGE IS  
OF POOR QUALITY



3. THIS TRANSFORM MAY BE USED TO SHIFT A SUBSTRING WITHIN A STRING BY USING THE SAME ACTUAL ARGUMENT FOR ISTOUT AND ISTIN AND BY SPECIFYING OUTPUT AND INPUT SUBSTRINGS WHICH PARTIALLY OVERLAP. PROVIDED THE DIRECTION OF THE SHIFT IS OPPOSITE TO THE DIRECTION OF THE SUBSTRINGS.
4. WHEN SHIFTING A SUBSTRING TO THE LEFT (LOCOUT<LOCIN) AND PADDING AT THE RIGHT, THEN BOTH LENOUT AND LENIN MUST BE POSITIVE (DIRECTED TO THE RIGHT) OR THE RESULTS WILL BE UNDEFINED.
5. WHEN SHIFTING A SUBSTRING TO THE RIGHT (LOCOUT>LOCIN) AND PADDING AT THE LEFT, THEN BOTH LENOUT AND LENIN MUST BE NEGATIVE (DIRECTED TO THE LEFT) OR THE RESULTS WILL BE UNDEFINED.
6. IF EITHER SUBSTRING LIES WHOLLY OR PARTIALLY OUTSIDE ITS STRING THE RESULTS WILL BE UNDEFINED.
7. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

GLOBAL DECLARATIONS  
-----

AKRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

LOCAL DECLARATIONS  
-----

NONE.

PROCEDURE  
-----

0(00) . 1-BANK . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY  
NOVIST:

COMPUTE INITIAL WORD ADDRESS FOR INPUT SUBSTRING

|       |           |                                  |
|-------|-----------|----------------------------------|
| LA    | AO.*4.X11 | . INTEGER LOCIN. COUNTING FROM 1 |
| AA.XU | AO.-1     | . INTEGER LOCIN. COUNTING FROM 0 |
| JN    | AO.RETURN | . RETURN IF TO LEFT OF STRING    |
| AA    | AO.3.X11  | . AO := WORD ADDRESS IN          |

COMPUTE INITIAL WORD ADDRESS FOR OUTPUT SUBSTRING

|       |           |                                   |
|-------|-----------|-----------------------------------|
| LA    | AO.*1.X11 | . INTEGER LOCOUT. COUNTING FROM 1 |
| AA.XU | AO.-1     | . INTEGER LOCOUT. COUNTING FROM 0 |
| JN    | AO.RETURN | . RETURN IF TO LEFT OF STRING     |
| AA    | AO.0.X11  | . AO := WORD ADDRESS OUT          |

DETERMINE INCREMENTATION & NUMBER OF WORDS FOR INPUT SUBSTRING

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

MOVIST  
003

```

.
LENIND      LA      A4.*5.X11      . LENIND
           JN      A4.INMINUS      .
INPLUS     LX1.XU   A0.*1          . WORD INCREMENTATION IS +1
           J        LENOUT         .
INMINUS    LX1.XU   A0.-1          . WORD INCREMENTATION IS -1
           LNA      A4.A4          . A4 := # OF WORDS IN
.
. DETERMINE INCREMENTATION & NUMBER OF WORDS FOR OUTPUT SUBSTRING
.
LENOUT      LA      A5.*2.X11      . LENOUT
           JN      A5.OUTMINUS     .
OUTPLUS    LX1.XU   A2.*1          . WORD INCREMENTATION IS +1
           J        MOVNPAD        .
OUTMINUS   LX1.XU   A2.-1          . WORD INCREMENTATION IS -1
           LNA      A5.A5          . A5 := # OF WORDS OUT
.
. INITIALIZE REPEAT REGISTERS. MOVE AND PAD
.
MOVNPAD     TO      A4.A5          . SKIP NEXT INSTR IF A5 > A4
           LA      A4.A5          . A4 := # OF WORDS TO MOVE
           ANA     A5.A4          . A5 := # OF WORDS TO PAD
MOVE        LR      R1.A4          . REPEAT COUNT FOR MOVE
           BT      A2.0.*A0       . MOVE WDS FROM ADDR IN A0 TO ADDR IN A2
PAD         LA      A0.6.X11      . A0 := ADDR OF PAD WORD
           LX1.U   A0.0           . NO INCREMENTATION
           LR      R1.A5          . REPEAT COUNT FOR PAD
           BT      A2.0.*A0       . PAD WORDS
.
. TERMINATION CODE
.
RETURN      J        0.X11        .
.
           END

```

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**NO4NI  
001**

.     **INTEGER FUNCTION**  
.     **0            8 NUMBER OF BYTES FOR NUMBER OF INTEGERS**  
.     **= NB4NI(**  
.     **1 NINTER) 8 NUMBER OF INTEGERS**  
.     -----  
.     .

.     **HISTORY**  
.     -----  
.     .

.     **MARY TOMPKINS       LEC       09/11/79       REQUIREMENTS**  
.     **CHARLES HELMKE     LEC       09/11/79       ALGORITHM DESIGN**  
.     **JIM CRISP           LEC       09/12/79       ALGORITHM CODE**  
.     .

.     **METHOD**  
.     -----  
.     .

.     **INPUT NUMBER OF INTEGERS. MULTIPLY BY 4.   OUTPUT NUMBER OF BYTES.**  
.     .

.     **MACHINE-DEPENDENT CODE**  
.     -----  
.     .

.     **WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 9-BIT**  
.     **QUARTER WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.**  
.     **BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.**  
.     **IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.**  
.     **DIFFERENT COMPILERS (EG. UNIVAC ASCII FORTRAN). AND DIFFERENT MACHINES.**  
.     .

.     **EXTERNAL REFERENCES**  
.     -----  
.     .

.     **NONE**  
.     .

.     **EXCEPTIONS**  
.     -----  
.     .

.     **1. IF NINTER IS LESS THAN 1, NB4NI IS SET EQUAL TO ZERO.**  
.     .

.     **GLOBAL DECLARATIONS**  
.     -----  
.     .

.     **AXRS       .   STANDARD UNIVAC 1100 REGISTER MNEMONICS**  
.     .

.     **LOCAL DECLARATIONS**  
.     -----  
.     .

.     **NONE.**  
.     .

.     **PROCEDURE**  
.     .

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

MS4N1  
002

```

. -----
.
S(00) . 1-BANK      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
MS4N1.      LA      A1,0.X11      . FETCH ARGUMENT
              SZ      A0      .
              JN      A1,RETURN      . IF ARGUMENT NEG RETURN ZERO
              LOSL     A0,30      . MULT A1 BY 4 AND STORE IN A0
RETURN      J      2.X11      .
              END

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

NC4NI  
001

. INTEGER FUNCTION  
. 0            8 NUMBER OF CHARACTERS FOR NUMBER OF INTEGERS  
. = NC4NI(  
. 1 NINTER) 8 NUMBER OF INTEGERS  
. -----

. HISTORY  
. -----

. JIM CRISP            LEC            09/11/79            REQUIREMENTS  
. MARY TOMPKINS        LEC            09/11/79            ALGORITHM DESIGN  
. CHARLES HELMKE       LEC            09/12/79            ALGORITHM CODE

. METHOD  
. -----

. INPUT NUMBER OF INTEGERS. MULTIPLY BY 8. OUTPUT NUMBER OF CHARACTERS.

. MACHINE-DEPENDENT CODE  
. -----

. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 6-BIT  
. FIELDATA CHARACTERS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
. BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.  
. DIFFERENT COMPILERS (EG. UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

. EXTERNAL REFERENCES  
. -----

. NONE

. EXCEPTIONS  
. -----

. 1. IF NINTER IS LESS THAN 1. NC4NI IS SET EQUAL TO ZERO.

. GLOBAL DECLARATIONS  
. -----

. AXRS            . STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS  
. -----

. NONE.

. PROCEDURE

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

NC4NI  
002

```

. -----
.
S(00) . 1-BANK      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
NC4NI*              LA      A1,0.X11      . FETCH ARGUMENT
                      S2      A0              .
                      JN      A1,RETURN     . IF ARGUMENT NEO RETURN ZERO
                      SA      A1,A0        .
                      NSI,U   A0,6         . MULT BY 6. STORE IN A0
RETURN              J      2.X11         .
                      END

```

```

SUBROUTINE NEXTOK( 8 GET POINTERS TO NEXT TOKEN IN IMAGE BUFFER
0 KHARI. 8 FIRST NON-'SPACE' CHARACTER OF NEXT TOKEN
0 LOCLN. 8 POINTERS: (1) LOCTOK: NEXT TOKEN LOCATION
C (2) LENTOK: NEXT TOKEN LENGTH
C (3) LENIMG: IMAGE BUFFER LENGTH ( *(-1) IF END )
C
1 LOCLEP. 8 POINTERS: (1) PREVIOUS TOKEN LOCATION
C (2) PREVIOUS TOKEN LENGTH
C (3) PREVIOUS IMAGE BUFFER LENGTH ( *(-1) IF END )
1 IMAGE. 8 IMAGE BUFFER CONTAINING TOKENS
1 KHRDLH. 8 DELIMITER CHARACTER (DELIMITER TOKENS ARE MADE UP OF THESE)
1 KHRSPA) 8 SPACE CHAR (LEADING SPACE CHARS IN NON-SPACE TOKEN ARE IGNORED
-----

```

C HISTORY  
C -----

```

C E M SCHLOSSER LEC 02/10/75 ORIGINAL CODE IN FIELDS & RITOL
C E M SCHLOSSER LEC 01/17/79 REVISE & GENERALIZE
C
C

```

C METHOD  
C -----

```

C THE STRING OF CHARACTERS IN THE IMAGE IS COMPRISED OF SUBSTRINGS OF TWO
C TYPES: DELIMITER TOKENS AND NON-DELIMITER TOKENS. A DELIMITER TOKEN IS
C A SUBSTRING MADE UP ENTIRELY OF DELIMITER CHARACTERS AND BOUNDED BY NON-
C DELIMITER CHARACTERS AND/OR IMAGE LIMITS. A NON-DELIMITER TOKEN IS A
C SUBSTRING (EXCLUDING LEADING 'SPACE' CHARACTERS) MADE UP ENTIRELY OF NON-
C DELIMITER CHARACTERS AND BOUNDED BY DELIMITER CHARACTERS AND/OR IMAGE
C LIMITS.
C
C TO LOCATE TOKENS IN SEQUENCE (ALTERNATING DELIMITER AND NON-DELIMITER)
C FROM AN IMAGE:
C 1. INITIALIZE LOCLEP(1) TO THE LOCATION WHERE SCANNING IS TO START
C (NORMALLY 1)
C 2. INITIALIZE LOCLEP(2) TO 0
C 3. INITIALIZE LOCLEP(3) TO THE LENGTH IN CHAR OF THE IMAGE BUFFER
C 4. CALL NEXTOK
C 5. IF LOCLN(3) IS NEGATIVE THEN THERE ARE NO MORE TOKENS AND:
C LOCLN(1) WILL POINT TO FIRST CHAR IN IMAGE (LOCLN(1)=1)
C LOCLN(2) WILL BE 0
C 6. OTHERWISE KHARI, LOCLN(1), LOCLN(2) THEN CONTAIN 1ST CHARACTER,
C LOCATION, LENGTH OF NEXT TOKEN
C 7. CONTINUE BY SETTING THE LOCLEP POINTERS TO THE LOCLN POINTERS AND
C CALLING NEXTOK UNTIL LOCLN(3) IS NEGATIVE.
C 8. TO ELIMINATE THE NEED FOR EXPLICIT UPDATING OF LOCLEP, THE SAME
C FORMAL ARGUMENT MAY BE USED FOR LOCLEP AND LOCLN.
C
C

```

C MACHINE-DEPENDENT CODE  
C -----

C NONE.

```

C
C EXTERNAL REFERENCES
C -----
C
C      GETCHR      & GET CHARACTER FROM CHARACTER STRING
C      INTEGER LCHREQ  & LOCATE CHARACTER IN SUBSTRING EQUAL TO SEARCH CHAR
C      INTEGER LCHRE  & LOCATE CHARACTER IN SUBSTRING NOT EQUAL TO SEARCH CHAR
C
C
C EXCEPTIONS
C -----
C
C      1. IF LOCLN(3) IS LESS THAN +1 ON ENTRY, THEN ALL ARGUMENTS ARE
C         UNCHANGED.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER LOCLEP(3)      & ARGUMENT
C      INTEGER LOCLN(3)      & ARGUMENT
C      DEFINE LOCTOK=LOCLN(1) & NEXT TOKEN LOCATION
C      DEFINE LENTOK=LOCLN(2) & NEXT TOKEN LENGTH
C      DEFINE LENIMG=LOCLN(3) & NEXT IMAGE BUFFER LENGTH ( +1-1) IF END IMAGE
C      INTEGER IMAGE(1)      & ARGUMENT
C      INTEGER LOCLND      & LOCATION AFTER LAST CHARACTER IN TOKEN
C
C
C PROCEDURE
C -----
C
C INITIALIZE OUTPUT LOCLN POINTERS
C
C      LOCLN(1)=LOCLEP(1)
C      LOCLN(2)=LOCLEP(2)
C      LOCLN(3)=LOCLEP(3)
C
C
C DETERMINE LOCATION OF NEXT TOKEN
C
C      IF(LENIMG.LE.0) GO TO 900      & NO IMAGE
C      LOCTOK=LOCTOK+LENTOK      & LOC AFTER PREVIOUS TOKEN
C      LENTOK=LENIMG-LOCTOK+1      & LENGTH OF REMAINING IMAGE
C      IF(LENTOK.LE.0) GO TO 900
C
C
C DETERMINE STATE OF NEXT TOKEN
C
C      CALL GETCHR(KHAR1,  IMAGE,LOCTOK)

```



DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

NEXTOK  
003

IF(KHARI.EQ.KHRDLN) 00 TO 400  
00 TO 300

C

C

C NON-DELIMITER TOKEN

C

300 LOCEND=LCHREQ(IMAGE.LOCTOK.LENTOK.KHRDLN) & LOC OF NEXT DELIMITER CHAR  
IF(LOCEND.EQ.0) LOCEND=LENIMG+1  
LENTOK=LOCEND-LOCTOK  
LOCTOK=MAX0(LOCTOK.LCHRNE(IMAGE.LOCTOK.LENTOK.KHRSPA)) & 1ST NON-SPACE  
LENTOK=LOCEND-LOCTOK  
CALL GETCHR(KHARI, IMAGE.LOCTOK)  
00 TO 900

C

C

C DELIMITER TOKEN

C

400 LOCEND=LCHRNE(IMAGE.LOCTOK.LENTOK.KHRDLN) & LOC AFTER LAST DELIMITER  
IF(LOCEND.EQ.0) LOCEND=LENIMG+1  
LENTOK=LOCEND-LOCTOK  
00 TO 900

C

C

C NO MORE TOKENS

C

800 LOCTOK=1  
LENTOK=0  
LENIMG=-1ABS(LENIMG)  
00 TO 900

C

C

C DONE

C

900 RETURN  
END

. INTEGER FUNCTION  
 . 0            8 NUMBER OF INTEGERS FOR NUMBER OF BYTES  
 . = N14NB(  
 . 1 NBYTES)    8 NUMBER OF BYTES  
 . -----

. HISTORY  
 . -----

|                |     |          |                  |
|----------------|-----|----------|------------------|
| CHARLES MELNKE | LEC | 09/11/79 | REQUIREMENTS     |
| JIM CRISP      | LEC | 09/11/79 | ALGORITHM DESIGN |
| MARY TOMPKINS  | LEC | 09/12/79 | ALGORITHM CODE   |

. METHOD  
 . -----

.        \*INTEGERS := (NBYTES/4) (+ 1 IF REMAINDER)

. MACHINE-DEPENDENT CODE  
 . -----

. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 9-BIT  
 . QUARTER WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
 . BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
 . IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.  
 . DIFFERENT COMPILERS (E.G. UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

. EXTERNAL REFERENCES  
 . -----

. NONE

. EXCEPTIONS  
 . -----

. 1. IF NBYTES IS LESS THAN 1, N14NB IS SET EQUAL TO ZERO.

. GLOBAL DECLARATIONS  
 . -----

.        AXRS        . STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS  
 . -----

. NONE.

. PROCEDURE

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**N14ND  
002**

. -----

**S(00) . 1-BANK  
N14ND**

**LA  
S2  
JN  
LOSL  
JZ  
AA.U**

**. BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY**

**A1.\*0.X11 . FETCH ARGUMENT  
A0 .  
A1.RETURN . IF ARGUMENT NEG RETURN ZERO  
A0.34 . A0 := A1/4  
A1.RETURN . IF NO REMAINDER. THEN RETURN  
A0.1 . REMAINDER -- ROUND UP**

**RETURN**

**J  
END**

**2.X11**

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**NI4NC  
001**

**INTEGER FUNCTION**  
**0**            **8** NUMBER OF INTEGERS FOR NUMBER OF CHARACTERS  
**NI4NC(**  
**1 NCHAR)**    **8** NUMBER OF CHARACTERS  
 -----

**HISTORY**  
 -----

|                |     |          |                  |
|----------------|-----|----------|------------------|
| CHARLES HELMKE | LEC | 08/11/79 | REQUIREMENTS     |
| JIM CRISP      | LEC | 08/11/79 | ALGORITHM DESIGN |
| MARY TOMPKINS  | LEC | 08/12/79 | ALGORITHM CODE   |

**METHOD**  
 -----

**NUMINTEGERS := (NUMCHARACTERS/8) + 1 IF REMAINDER**

**MACHINE-DEPENDENT CODE**  
 -----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 8-BIT  
 FIELDATA CHARACTERS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
 BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
 IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.  
 DIFFERENT COMPILERS (EO, UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

**EXTERNAL REFERENCES**  
 -----

**NONE**

**EXCEPTIONS**  
 -----

**1. IF NCHAR IS LESS THAN 1, NI4NC IS SET EQUAL TO ZERO.**

**GLOBAL DECLARATIONS**  
 -----

**AXRS**            **STANDARD UNIVAC 1100 REGISTER MNEMONICS**

**LOCAL DECLARATIONS**  
 -----

**NONE.**

**PROCEDURE**

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

NI4NC  
002

```

. -----
.
S(00) . 1-BANK      . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY
NI4NC.             LA  A1.0.X11      . FETCH ARGUMENT
                   S2  A0              .
                   JN  A1.RETURN      . IF ARGUMENT NEG RETURN ZERO
                   DI.U A0.0          . A0 := A1/8
                   JZ  A1.RETURN      . IF NO REMAINDER, THEN RETURN
                   AA.U A0.1          . REMAINDER -- ROUND UP

RETURN             J    2.X11
                   END

```

SUBROUTINE PUTBYT: 8 PUT NON-NEG INTEGER INTO BYTE OF BYTE STRING  
0 IBYSTR, 8 INTERNAL BYTE STRING  
1 IBYLOC, 8 BYTE LOCATION WITHIN STRING (COUNTING FROM 1 AT LEFT)  
1 IBYT) 8 UNPACKED BYTE (RIGHT-ALIGNED)  
-----

HISTORY  
-----

|               |     |          |                    |
|---------------|-----|----------|--------------------|
| E H SCHLOSSER | LEC | 07/07/70 | ORIGINAL CODE      |
| E H SCHLOSSER | LEC | 07/11/70 | SUPPORT MINUS ZERO |

METHOD  
-----

CONVERT ARRAY-RELATIVE QUARTER-WORD LOCATION TO MACHINE-DEPENDENT  
LOCATION AS FOLLOWS:  
WORD NUMBER = 34 MOST SIGNIFICANT BITS  
QUARTER-WORD LOCATION WITHIN WORD = 2 LEAST SIGNIFICANT BITS  
STORE BYTE IN 8-BIT QUARTER WORD

MACHINE-DEPENDENT CODE  
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 8-BIT  
QUARTER WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES,  
DIFFERENT COMPILERS (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

EXTERNAL REFERENCES  
-----

NONE

EXCEPTIONS  
-----

1. THE COMPUTER MUST BE IN QUARTER-WORD MODE. (ASSEMBLE WITH BASH.F  
OR COLLECT WITH SHAP.F OR INSURE THAT SETQWD WAS CALLED PREVIOUSLY.)
2. IBYSTR IS UNCHANGED IF IBYLOC IS LESS THAN 1.
3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
4. RESULTS ARE ONLY DEFINED FOR 0 <= IBYT <= 255.
5. MINUS ZERO IS AUTOMATICALLY CONVERTED TO PLUS ZERO BEFORE INSERTION  
IN STRING.

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

PUTBYT  
002

. GLOBAL DECLARATIONS  
. -----

AKRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS  
. -----

NONE.

. PROCEDURE  
. -----

|                |       |           |                                                         |
|----------------|-------|-----------|---------------------------------------------------------|
| 01001 . 1-BANK |       |           | . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY |
| PUTBYT:        | LA    | A1.*1.X11 | . BYTE LOCATION. COUNTING FROM 1                        |
|                | AA.XU | A1.-1     | . BYTE LOCATION. COUNTING FROM 0                        |
|                | LMA   | A3.*2.X11 | . BYTE CONTENTS                                         |
|                | JN    | A1.RETURN | . LOCATION TO LEFT OF ARRAY!!!                          |
|                | DSL   | A1.2      | . A1 = WORD LOCATION                                    |
|                | SSL   | A2.34     | . A2 = QWD LOCATION WITHIN WORD                         |
|                | AA    | A1.0.X11  | . CONVERT WORD LOCATION TO ADDRESS                      |
|                | EX    | INSERT.A2 | . INSERT BYTE INTO PROPER QTR                           |
| RETURN         | J     | 4.X11     | .                                                       |
| INSERT         | SA.Q1 | A3.0.A1   | .                                                       |
|                | SA.Q2 | A3.0.A1   | .                                                       |
|                | SA.Q3 | A3.0.A1   | .                                                       |
|                | SA.Q4 | A3.0.A1   | .                                                       |
|                | END   |           |                                                         |

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**PUTCHR  
001**

.     SUBROUTINE PUTCHR(    PUT ONE CHARACTER INTO CHARACTER STRING  
.     0 JSTRNG.    CHARACTER STRING  
.     1 KHRLOC.    CHARACTER LOCATION WITHIN STRING (COUNTING FROM 1 AT LEFT)  
.     .     .  
.     1 KHR)        UNPACKED CHARACTER (LEFT-ALIGNED. REST OF WORD IGNORED)  
.     -----

.     ENTRY PUTICE(    PUT INTEGER CHARACTER EQUIVALENT INTO CHARACTER STRING  
.     0 JSTRNG.    CHARACTER STRING  
.     1 KHRLOC.    CHARACTER LOCATION WITHIN STRING (COUNTING FROM 1 AT LEFT)  
.     .     .  
.     1 NICE)        INTEGER CHAR EQUIVALENT (RIGHT-ALIGNED. REST OF WD IGNORED)  
.     -----

.     HISTORY  
.     -----

.           E M SCHLOSSER    LEC    08/05/78    ORIGINAL CODE  
.           E M SCHLOSSER    LEC    07/27/79    SUPPORT MINUS ZERO IN PUTICE

.     METHOD  
.     -----

.     CONVERT STRING-RELATIVE CHARACTER LOCATION TO MACHINE-DEPENDENT  
.     LOCATION AS FOLLOWS:  
.           WORD NUMBER = (CHARACTER LOCATION - 1) / CHARACTERS PER WORD  
.           CHARACTER LOCATION WITHIN WORD = REMAINDER FROM DIVISION  
.     STORE CHARACTER IN 8-BIT SIXTH WORD

.     MACHINE-DEPENDENT CODE  
.     -----

.     WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 8-BIT  
.     FIELDATA CHARACTERS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
.     BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
.     IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT CHARACTER CODES.  
.     DIFFERENT COMPILERS (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

.     EXTERNAL REFERENCES  
.     -----

.     NONE

.     EXCEPTIONS  
.     -----

- .     1. JSTRNG IS UNCHANGED IF KHRLOC IS LESS THAN 1.
- .     2. RESULTS ARE ONLY DEFINED FOR 0 <= NICE <= +83.



DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

PUTCHR  
002

3. MINUS ZERO IS AUTOMATICALLY CONVERTED TO PLUS ZERO BEFORE  
INSERTION IN STRING.

4. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

GLOBAL DECLARATIONS

AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

LOCAL DECLARATIONS

NONE.

PROCEDURE

S(00) . 1-BANK . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY

|         |            |                      |                                    |
|---------|------------|----------------------|------------------------------------|
| PUTCHR* | LA.S1<br>J | A3.*2.X11<br>GETADDR | . LOAD LEFT-ALIGNED CHARACTER      |
| PUTICE* | LMA        | A3.*2.X11            | . LOAD ABSOLUTE VALUE OF ICE       |
| GETADDR | LA         | A2.*1.X11            | . CHAR LOCATION. COUNTING FROM 1   |
|         | AA.XU      | A2.-1                | . CHAR LOCATION. COUNTING FROM 0   |
|         | SZ         | A1                   | . DIVISION COMING                  |
|         | JN         | A2.RETURN            | . LOCATION TO LEFT OF STRING!!!    |
|         | DI.XU      | A1.6                 | . A1=WORD LOC. A2=SIXTH LOC        |
|         | AA         | A1.0.X11             | . CONVERT WORD LOCATION TO ADDRESS |
|         | EX         | INSERT.A2            | . INSERT CHAR INTO PROPER SIXTH    |
| RETURN  | J          | *.X11                | .                                  |
| INSERT  | SA.S1      | A3.0.A1              | .                                  |
|         | SA.S2      | A3.0.A1              | .                                  |
|         | SA.S3      | A3.0.A1              | .                                  |
|         | SA.S4      | A3.0.A1              | .                                  |
|         | SA.S5      | A3.0.A1              | .                                  |
|         | SA.S6      | A3.0.A1              | .                                  |

END

. SUBROUTINE PUTDBY( 8 PUT NON-NEG INTEGER INTO DOUBLE BYTE OF BYTE STRING  
. 0 IBYSTR, 8 INTERNAL BYTE STRING  
. 1 IBYLOC, 8 BYTE LOCATION WITHIN STRING OF FIRST BYTE IN DOUBLE BYTE  
. 8 (COUNTING FROM 1 AT LEFT OF BYTE STRING)  
. 1 IBYT) 8 UNPACKED DOUBLE BYTE (RIGHT-ALIGNED)  
-----

. HISTORY  
-----

. E H SCHLOSSER LEC 07/09/79 ORIGINAL CODE

. METHOD  
-----

. CONVERT ARRAY-RELATIVE QUARTER-WORD LOCATION TO MACHINE-DEPENDENT  
. LOCATION AS FOLLOWS:  
. WORD NUMBER = 34 MOST SIGNIFICANT BITS  
. QUARTER-WORD LOCATION WITHIN WORD = 2 LEAST SIGNIFICANT BITS  
. SPLIT INTO TWO 8-BIT BYTES & STORE INTO TWO 9-BIT QUARTER WORDS

. MACHINE-DEPENDENT CODE  
-----

. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 9-BIT  
. QUARTER WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT.  
. BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES.  
. DIFFERENT COMPILERS (EG.. UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

. EXTERNAL REFERENCES  
-----

. NONE

. EXCEPTIONS  
-----

- . 1. THE COMPUTER MUST BE IN QUARTER-WORD MODE. (ASSEMBLE WITH 8ASH.F  
. OR COLLECT WITH 8MAP.F OR INSURE THAT SETQWD WAS CALLED PREVIOUSLY.)
- . 2. IBYSTR IS UNCHANGED IF IBYLOC IS LESS THAN 1.
- . 3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
- . 4. RESULTS ARE ONLY DEFINED FOR 0 <= IBYT <= +65,535.
- . 5. MINUS ZERO IS AUTOMATICALLY CONVERTED TO PLUS ZERO BEFORE INSERTION  
. IN STRING.

. GLOBAL DECLARATIONS  
-----

AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS  
-----

NONE.

. PROCEDURE  
-----

|                |       |                                                         |
|----------------|-------|---------------------------------------------------------|
| S(00) . 1-BANK |       | . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY |
| PUTBYT:        | LA    | A0.*1.X11 . BYTE LOCATION. COUNTING FROM 1              |
|                | AA.XU | A0.-1 . BYTE LOCATION. COUNTING FROM 0                  |
|                | LMA   | A3.*2.X11 . DOUBLE BYTE CONTENTS                        |
|                | JN    | A0.RETURN . LOCATION TO LEFT OF ARRAY!!!                |
|                | DSL   | A0.2 . A0 = WORD LOCATION                               |
|                | SSL   | A1.34 . A1 = QWD LOCATION WITHIN WORD                   |
|                | AA    | A0.0.X11 . CONVERT WORD LOCATION TO ADDRESS             |
|                | LDSL  | A2.28 . SHIFT FIRST 8 BITS INTO A2                      |
|                | EX    | PUTBYT1.A1 . PUT INTO FIRST BYTE                        |
|                | LA.U  | A2.0 . CLEAR A2                                         |
|                | LDSL  | A2.8 . SHIFT SECOND 8 BITS INTO A2                      |
|                | EX    | PUTBYT2.A1 . PUT INTO SECOND BYTE                       |
| RETURN         | J     | 4.X11 .                                                 |
| PUTBYT1        | SA.Q1 | A2.0.A0 .                                               |
| PUTBYT2        | SA.Q2 | A2.0.A0 .                                               |
|                | SA.Q3 | A2.0.A0 .                                               |
|                | SA.Q4 | A2.0.A0 .                                               |
|                | SA.Q1 | A2.1.A0 .                                               |
|                | END   |                                                         |

```

SUBROUTINE PUTHEX (      & PUT HEXADECEMAL CHAR IN NYBLE IN BYTE STRING
0 IDYSTR.      & BYTE STRING
1 IDYLOC.      & LOCATION OF SUBSTRING WITHIN STRING
1 NYBLOC.      & NYBLE LOCATION WITHIN SUBSTRING
C              (BOTH COUNTING FROM 1 AT LEFT OF STRING/SUBSTRING)
C
C      I NYBHEX)      & HEX DIGIT EQUIVALENT OF NYBLE VALUE
C -----
C
C
C HISTORY
C -----
C
C      J C CRISP      LEC      07/25/79      REQUIREMENTS
C      J C CRISP      LEC      08/24/79      ALGORITHM DESIGN
C      J C CRISP      LEC      08/24/79      ALGORITHM CODING
C
C
C METHOD
C -----
C
C      IF NYBHEX IS '0' THROUGH '9'. THEN NYBLE=ICE(NYBHEX)-ICE('0').
C      OTHERWISE NYBLE=ICE(NYBHEX)-ICE('A')+10. CALL PUTNYB TO PUT THAT
C      VALUE INTO THE NYBLE.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE
C
C EXTERNAL REFERENCES
C -----
C
C      PUTNYB      & PUT NON-NEGATIVE INTEGER INTO NYBLE OF BYTE STRING
C      INTEGER ICE      & INTEGER CHARACTER EQUIVALENT (FOR CHARACTER)
C
C EXCEPTIONS
C -----
C
C      1. RESULTS ARE ONLY DEFINED FOR NYBHEX WITH A VALUE IN THE RANGE
C      OF HEXADECEMAL DIGITS ('0'-'9' AND 'A'-'F').
C
C      2. IDYSTR IS UNCHANGED IF IDYLOC AND NYBLOC REFER TO A NYBLE TO
C      THE LEFT OF THE BYTE STRING.
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE
C
C LOCAL DECLARATIONS

```

DAN PACKAGE APPENDIX R  
 CHAR/BYTE/STRING ROUTINES

PUTHEX  
 002

```

C -----
C
C      INTEGER NYBLE      & NON-NEGATIVE INTEGER VALUE OF NYBLE
C      INTEGER NYBMEX     & I-C-E OF HEX DIGIT
C
C
C PROCEDURE
C -----
C
C      ICEMEX=ICE(NYBMEX)
C
C
C CHECK IF CHARACTER IN RANGE '0'-'9'--SET NYBLE
C
C      IF (ICEMEX.GE.ICE('0').AND.ICEMEX.LE.ICE('9')) GO TO 100
C
C
C ELSE NYBLE IN RANGE 'A'-'F'--SET NYBLE
C
C      NYBLE=(ICEMEX-ICE('A'))*10
C      GO TO 500
C
C
C 100 NYBLE=ICEMEX-ICE('0')
C
C
C PUT NYBLE IN BYTE STRING
C
C 500 CALL PUTNYB (IBYSTR,(IBYLOC),(NYBLOC), NYBLE)
C
C
C      RETURN
C      END
  
```

. SUBROUTINE PUTINT( 8 PUT INTEGER INTO INTEGER STRING  
. 0 INTSTR, 8 INTEGER STRING  
. ( INTLOC, 8 INTEGER LOCATION WITHIN STRING (COUNTING FROM 1 AT LEFT)  
. 1  
. 1 INTEGER, 8 INTEGER  
-----

. HISTORY  
-----

. E H SCHLOSSER LEC 10/18/79 REQUIREMENTS  
. E H SCHLOSSER LEC 10/19/79 DESIGN & CODE

. METHOD  
-----

. CONVERT STRING-RELATIVE INTEGER LOCATION TO MACHINE-DEPENDENT  
. ADDRESS. THEN INSERT INTEGER INTO THAT ADDRESS.  
. THIS TRANSFORM IS PROVIDED FOR COMPATIBILITY WITH PUTBYT AND PUTCHR.

. MACHINE-DEPENDENT CODE  
-----

. WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS.  
. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT, BUT  
. THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
. IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES,  
. DIFFERENT COMPILERS (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

. EXTERNAL REFERENCES  
-----

. NONE

. EXCEPTIONS  
-----

- . 1. INTSTR IS UNCHANGED IF INTLOC IS LESS THAN 1.
- . 2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

. GLOBAL DECLARATIONS  
-----

. AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS  
-----

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

PUTINT  
002

. NONE.

. PROCEDURE

. -----

S(00) . 1-BANK

PUTINT\*

LA

AA.XU

JN

AA

LA

SA

. BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY

A1.\*1.X11

. INTEGER LOCATION, COUNTING FROM 1

A1.-1

. INTEGER LOCATION, COUNTING FROM 0

A1.RETURN

. LOCATION TO LEFT OF STRING!!!

A1.0.X11

. CONVERT LOCATION TO ADDRESS

A2.\*2.X11

. A2 := INTGER

A2.0.A1

. INTSTR(INTLOC) := A2

. RETURN

J

4.X11

END

SUBROUTINE PUTNYB( 8 PUT NON-NEG INTEGER INTO NYBLE OF BYTE STRING  
0 IBYSTR. 8 INTERNAL BYTE STRING  
( IBYLOC. 8 BYTE LOCATION OF SUBSTRING WITHIN STRING  
( NYBLOC. 8 NYBLE LOCATION OF NYBLE WITHIN SUBSTRING  
8 (BOTH COUNTING FROM 1 AT LEFT OF STRING/SUBSTRING)  
1 NYBLE) 8 UNPACKED NYBLE (RIGHT-ALIGNED)  
-----

HISTORY  
-----

E H SCHLOSSER LEC 07/10/79 ORIGINAL CODE

METHOD  
-----

CONVERT STRING-RELATIVE NYBLE LOCATION TO MACHINE-DEPENDENT  
LOCATION AS FOLLOWS:

STRING-RELATIVE LOCATION IN NYBLES = 2\*IBYLOC+NYBLOC  
WORD NUMBER = 33 MOST SIGNIFICANT BITS  
NYBLE LOCATION WITHIN WORD = 3 LEAST SIGNIFICANT BITS  
MASK OUT OLD CONTENTS OF NYBLE & 'OR' IN NEW CONTENTS

MACHINE-DEPENDENT CODE  
-----

WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 9-BIT  
QUARTER WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT,  
BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.  
IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES,  
DIFFERENT COMPILERS (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

EXTERNAL REFERENCES  
-----

NONE

EXCEPTIONS  
-----

1. THE COMPUTER MUST BE IN QUARTER-WORD MODE. (ASSEMBLE WITH BASH.F  
OR COLLECT WITH SHAP.F OR INSURE THAT SETQWD WAS CALLED PREVIOUSLY.)
2. IBYSTR IS UNCHANGED IF IBYLOC & NYBLOC REFER TO A NYBLE TO THE  
LEFT OF THE BYTE STRING.
3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
4. RESULTS ARE ONLY DEFINED FOR 0 <= NYBLE <= 15.



**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**PUTNYB  
002**

. S. MINUS ZERO IS AUTOMATICALLY CONVERTED TO PLUS ZERO BEFORE INSERTION  
IN STRING.

. GLOBAL DECLARATIONS  
-----

AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS  
-----

NONE.

. PROCEDURE  
-----

|                |       |                                                         |                                            |
|----------------|-------|---------------------------------------------------------|--------------------------------------------|
| S(00) . 1-BANK |       | . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY |                                            |
| PUTNYB*        | LA    | A0.*1.X11                                               | . BYTE LOC OF SUBSTRING. COUNTING FROM 1   |
|                | LSSL  | A0.1                                                    | . NYBLE LOC OF SUBSTRING. COUNTING FROM 2  |
|                | AA    | A0.*2.X11                                               | . NYBLE LOC OF NYBLE. COUNTING FROM 3      |
|                | AA.XU | A0.-3                                                   | . NYBLE LOC OF NYBLE. COUNTING FROM 0      |
|                | JN    | A0.RETURN                                               | . LOCATION TO LEFT OF STRING!!!            |
|                | OSL   | A0.3                                                    | . A0 := WORD LOCATION WITHIN STRING        |
|                | SSL   | A1.33                                                   | . A1 := NYBLE LOCATION WITHIN WORD         |
|                | AA    | A0.0.X11                                                | . A0 := WORD ADDRESS                       |
|                | LA.XU | A2.-15                                                  | . MASK WITH ALL BITS IN NYBLE OFF          |
|                | EX    | ALION.A1                                                | . SHIFT MASK INTO NYBLE LOCATION WITHIN A2 |
|                | AND   | A2.0.A0                                                 | . MASK OUT OLD CONTENTS OF NYBLE ...       |
|                | SA    | A3.R1                                                   | . ... & STORE MASKED WORD                  |
|                | LMA   | A2.*3.X11                                               | . NEW CONTENTS OF NYBLE                    |
|                | EX    | ALION.A1                                                | . SHIFT CONTENTS INTO NYBLE LOC WITHIN A2  |
|                | OR    | A2.R1                                                   | . INSERT NEW CONTENTS OF NYBLE ...         |
|                | SA    | A3.0.A0                                                 | . ... & STORE WORD BACK INTO STRING        |
| RETURN         | J     | 5.X11                                                   | .                                          |
| ALION          | LSSC  | A2.31                                                   | . NYBLE 1 OF WORD                          |
|                | LSSC  | A2.27                                                   | . NYBLE 2 OF WORD                          |
|                | LSSC  | A2.23                                                   | . NYBLE 3 OF WORD                          |
|                | LSSC  | A2.19                                                   | . NYBLE 4 OF WORD                          |
|                | LSSC  | A2.15                                                   | . NYBLE 5 OF WORD                          |
|                | LSSC  | A2.11                                                   | . NYBLE 6 OF WORD                          |
|                | LSSC  | A2.7                                                    | . NYBLE 7 OF WORD                          |
|                | NOP   |                                                         | . NYBLE 8 OF WORD                          |
|                | END   |                                                         |                                            |

```

SUBROUTINE PUTQBY: 8 PUT INTEGER INTO QUADRUPLE BYTE OF BYTE STRING
0 IBYSTR. 8 INTERNAL BYTE STRING
1 IBYLOC. 8 BYTE LOCATION WITHIN STRING OF FIRST BYTE IN QUADRUPLE BYTE
8 (COUNTING FROM 1 AT LEFT OF BYTE STRING)
.
.
1 IQBYT) 8 UNPACKED QUADRUPLE BYTE (RIGHT-ALIGNED)
-----

```

HISTORY  
-----

```

.      E M SCHLOSSER      LEC      07/09/79      ORIGINAL CODE
.      E M SCHLOSSER      LEC      07/27/79      SUPPORT MINUS ZERO

```

METHOD  
-----

```

.      CONVERT ARRAY-RELATIVE QUARTER-WORD LOCATION TO MACHINE-DEPENDENT
.      LOCATION AS FOLLOWS:
.          WORD NUMBER = 34 MOST SIGNIFICANT BITS
.          QUARTER-WORD LOCATION WITHIN WORD = 2 LEAST SIGNIFICANT BITS
.      SPLIT INTO FOUR 8-BIT BYTES & STORE INTO FOUR 9-BIT QUARTER WORDS

```

MACHINE-DEPENDENT CODE  
-----

```

.      WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS USING 9-BIT
.      QUARTER WORDS. THE ARGUMENTS ARE DESIGNED TO BE MACHINE-INDEPENDENT,
.      BUT THE METHOD OF PASSING ARGUMENTS IS THAT USED BY UNIVAC FORTRAN V.
.      IMPLEMENTING CODE MUST BE REWRITTEN FOR DIFFERENT WORD SIZES,
.      DIFFERENT COMPILERS (E.G., UNIVAC ASCII FORTRAN), AND DIFFERENT MACHINES.

```

EXTERNAL REFERENCES  
-----

```

.      NONE

```

EXCEPTIONS  
-----

1. THE COMPUTER MUST BE IN QUARTER-WORD MODE. (ASSEMBLE WITH BASH.F  
OR COLLECT WITH SNAP.F OR INSURE THAT SETQWD WAS CALLED PREVIOUSLY.)
2. IBYSTR IS UNCHANGED IF IBYLOC IS LESS THAN 1.
3. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
4. RESULTS ARE UNDEFINED IF ANY EXCEPT THE 32 LEAST SIGNIFICANT BITS  
OF IQBYT ARE NON-ZERO.
5. MINUS ZERO IS AUTOMATICALLY CONVERTED TO PLUS ZERO BEFORE INSERTION

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**PUTBYT  
002**

. IN STRING.

. GLOBAL DECLARATIONS

. -----

. AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS

. LOCAL DECLARATIONS

. -----

. NONE.

. PROCEDURE

. -----

```

S100) . 1-BANK
PUTBYT:  LA      A0.01.X11      . BYTE LOCATION, COUNTING FROM 1
        AA.XU   A0.-1          . BYTE LOCATION, COUNTING FROM 0
        LA      A3.02.X11      . QUADRUPE BYTE CONTENTS
        JN      A0.RETURN      . LOCATION TO LEFT OF ARRAY!!!
        TNZ     A3              . IF PLUS OR MINUS ZERO, THEN ...
        LA.U    A3.0           . ... MAKE IT PLUS ZERO!
        DSL     A0.2            . A0 = WORD LOCATION
        SSL     A1.34           . A1 = QWD LOCATION WITHIN WORD
        AA      A0.0.X11       . CONVERT WORD LOCATION TO ADDRESS
        LDSL    A2.12          . SHIFT FIRST 8 BITS INTO A2
        EX      PUTBYT1.A1     . PUT INTO FIRST BYTE
        LA.U    A2.0            . CLEAR A2
        LDSL    A2.8           . SHIFT SECOND 8 BITS INTO A2
        EX      PUTBYT2.A1     . PUT INTO SECOND BYTE
        LA.U    A2.0            . CLEAR A2
        LDSL    A2.8           . SHIFT THIRD 8 BITS INTO A2
        EX      PUTBYT3.A1     . PUT INTO THIRD BYTE
        LA.U    A2.0            . CLEAR A2
        LDSL    A2.8           . SHIFT FOURTH 8 BITS INTO A2
        EX      PUTBYT4.A1     . PUT INTO FOURTH BYTE

RETURN  J        4.X11        .

PUTBYT1 SA.Q1     A2.0.A0      .
PUTBYT2 SA.Q2     A2.0.A0      .
PUTBYT3 SA.Q3     A2.0.A0      .
PUTBYT4 SA.Q4     A2.0.A0      .
        SA.Q1     A2.1.A0      .
        SA.Q2     A2.1.A0      .
        SA.Q3     A2.1.A0      .

```

END

. SUBROUTINE SETQWD 8 SET QUARTER WORD MODE  
.....  
.  
.  
.  
HISTORY  
.....  
.  
E M SCHLOSSER LEC 07/12/70 ORIGINAL CODE  
.  
.  
METHOD  
.....  
.  
EXECUTIVE REQUEST.  
.  
.  
MACHINE-DEPENDENT CODE  
.....  
.  
WRITTEN IN ASSEMBLER FOR THE UNIVAC 1100 SERIES COMPUTERS UNDER THE  
EXEC-8 OPERATING SYSTEM.  
.  
.  
EXTERNAL REFERENCES  
.....  
.  
ER PERS CHANGE BIT IN PROCESSOR STATE REGISTER  
.  
.  
EXCEPTIONS  
.....  
.  
1. THE COMPUTER MUST BE IN QUARTER-WORD MODE WHEN CALLING ANY TRANSFORMS  
WHICH PERFORM BYTE OR NYBLE OPERATIONS (GETBYT, PUTBYT, MOVBYT, ETC.)  
.  
2. QUARTER-WORD MODE MAY BE SET BY ANY OF THE FOLLOWING:  
CALL SETQWD WITHIN EXECUTING PROGRAM  
SASH.F WHEN ASSEMBLING SUB-PROGRAM  
SHAP.F WHEN COLLECTING PROGRAM  
.  
.  
GLOBAL DECLARATIONS  
.....  
.  
AXRS . STANDARD UNIVAC 1100 REGISTER MNEMONICS  
.  
.  
LOCAL DECLARATIONS  
.....  
.  
S(81) . D-BANK . BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY  
NF FORM 10.1.10.1  
MASK NF 0.1.0.1  
.  
.  
PROCEDURE

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

SETOND  
002

. -----

S(00) . I-BANK  
SETOND:

LA  
ER  
J

. BANKS REVERSED TO ELIMINATE INDIRECT LOAD TABLE ENTRY  
AG.MASK  
PERB  
I.XI:

END

**DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES**

**SLOST  
001**

**(NOT IMPLEMENTED)**

```

LOGICAL FUNCTION
0      & TRUE IF CHAR STRING IS ALPHABETIC (A-Z,SPACE). ELSE FALSE
= TRUAL(
I KSTRG.  & CHARACTER STRING
( KLOC.   & CHAR LOCATION WITHIN STRING WHERE SUBSTRING BEGINS
( LEND)   & LENGTH & DIRECTION OF SUBSTRING (POSITIVE = L TO R)
          (CHAR LOCATION COUNTED FROM 1 AT LEFT OF STRING)
-----
C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      02/12/79      ORIGINAL CODE
C
C
C METHOD
C -----
C
C      SCAN SUBSTRINGS. EXAMINING INTEGER-CHARACTER-EQUIVALENTS UNTIL THEY
C      ARE NON-ALPHABETIC OR UNTIL THE SUBSTRING IS EXHAUSTED.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      INTEGER ICE      & INTEGER CHARACTER EQUIVALENT FUNCTION FOR CHARACTER
C      GETICE          & GET ICE FROM STRINGS
C
C
C EXCEPTIONS
C -----
C
C      1. AN EMPTY CHARACTER STRING (LEND=0) IS NOT ALPHABETIC.
C
C      2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER INC      & SIGNED INCREMENTATION (+1 OR -1)
C      INTEGER LOCEND   & LOCATION OF LAST CHAR AT END OF SUBSTRING
C      INTEGER LOCTAL    & LOCATION OF 1-C-E BEING TESTED FOR ALPHABETIC

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

TRUAL  
002

```
      INTEGER ICETAL      & I-C-E BEING TESTED FOR ALPHABETIC
C
C
C PROCEDURE
C -----
C
C
C INITIALIZE SCAN DIRECTIONS & POINTERS
C
      TRUAL=.FALSE.
      IF(LEND.EQ.0) GO TO 900
      INC=1SIGN(1,LEND)
      LOCEND=KLOC+LEND-INC
C
C
C SCAN & TEST CHARACTERS
C
      DO 500 LOCTAL=KLOC,LOCEND,INC
      CALL GETICE(ICETAL, KSTRO,LOCTAL)
      IF(ICETAL.EQ.ICE(' ')) GO TO 500
      IF(ICETAL.LT.ICE('A')) GO TO 900
      IF(ICETAL.GT.ICE('Z')) GO TO 900
500 CONTINUE
      TRUAL=.TRUE.
C
C
C
C
C DONE
C
900 RETURN
      END
```



LOGICAL FUNCTION

0            8 TRUTH VALUE OF CHARACTER STRING COMPARISON  
- TRUCST:  
1 KSTRO1.    8 CHARACTER STRING 1  
( KLOC1.    8 CHAR LOCATION WITHIN STRING 1 WHERE SUBSTRING 1 BEGINS  
( LEND1.    8 LENGTH & DIRECTION OF SUBSTRING 1 (POSITIVE = L TO R)  
1 KOP.       8 COMPARISON OPERATOR ('<', '=', '>', '<=', '>=', '>=')  
1 KSTRO2.    8 CHARACTER STRING 2  
( KLOC2.    8 CHAR LOCATION WITHIN STRING 2 WHERE SUBSTRING 2 BEGINS  
( LEND2)    8 LENGTH & DIRECTION OF SUBSTRING 2 (POSITIVE = L TO R)  
             (CHAR LOCATION COUNTED FROM 1 AT LEFT OF STRING)

HISTORY

E H SCHLOSSER    LEC    06/27/78    ORIGINAL CODE

METHOD

SCAN SUBSTRINGS. COMPARING INTEGER-CHARACTER-EQUIVALENTS UNTIL THEY  
ARE UNEQUAL OR UNTIL BOTH SUBSTRINGS ARE EXHAUSTED. (IF ONE SUBSTRING  
IS EXHAUSTED, TREAT IT AS IF PADDED WITH SPACES.) USE THE RESULT FROM THE  
LAST I-C-E COMPARISON TO DETERMINE THE TRUTH VALUE OF THE RELATION BEING  
EVALUATED.

MACHINE-DEPENDENT CODE

NONE.

EXTERNAL REFERENCES

INTEGER ICE       8 INTEGER CHARACTER EQUIVALENT FUNCTION FOR CHARACTER  
OETICE            8 GET ICE FROM STRINGS

EXCEPTIONS

1. IF THERE IS NO CORRESPONDING CHARACTER IN A SUBSTRING, THE CHARACTER  
SPACE IS ASSUMED.
2. NO OTHER CHECKS ARE MADE ON ANY OF THE ARGUMENTS.

GLOBAL DECLARATIONS

NONE.

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

TRUCST  
002

```

C
C
C LOCAL DECLARATIONS
C -----
C
      INTEGER KP1,KP2           & CHAR LOCATIONS OF CHARACTERS BEING COMPARED
      INTEGER INC1,INC2        & SIGNED INCREMENTATION (+1 OR -1) FOR KP1,KP2
      INTEGER NCH1,NCH2        & NUMBER OF CHARACTERS IN SUBSTRINGS
      INTEGER NCHMAX           & NUMBER OF CHARACTERS IN LONGEST SUBSTRING
      INTEGER ICEPAD           & INTEGER-CHAR-EQUIV OF BLANK TO PAD SHORT SUBSTR
      INTEGER NCH              & NUMBER OF CHARACTERS COMPARED
      INTEGER NICE1,NICE2      & I-C-E'S OF CHARACTERS BEING COMPARED

C
C
C PROCEDURE
C -----
C
C INITIALIZE SCAN DIRECTIONS & POINTERS
C
      KP1=KLOC1
      INC1=ISIGN(1,LEND1)
      NCH1=IABS(LEND1)
      KP2=KLOC2
      INC2=ISIGN(1,LEND2)
      NCH2=IABS(LEND2)
      NCHMAX=MAX0(NCH1,NCH2)
      ICEPAD=ICE(' ')

C
C
C SCAN & COMPARE CHARACTERS
C
      DO 500 NCH=1,NCHMAX
        NICE1=ICEPAD
        IF(NCH.LE.NCH1) CALL GETICE(NICE1, KSTR01,KP1)
        KP1=KP1+INC1
        NICE2=ICEPAD
        IF(NCH.LE.NCH2) CALL GETICE(NICE2, KSTR02,KP2)
        KP2=KP2+INC2
        IF(NICE1-NICE2) 510, 500, 520
                        1<2 1=2 1>2
C
      500 CONTINUE
C
C
C NICE1 = NICE2
C
      TRUCST=((KOP.EQ.'=').OR.(KOP.EQ.'<=').OR.(KOP.EQ.'>='))
      GO TO 900
C
C
C NICE1 < NICE2
C
      510 TRUCST=((KOP.EQ.'<').OR.(KOP.EQ.'<=').OR.(KOP.EQ.'<>'))
      GO TO 900
C
C

```

DAN PACKAGE APPENDIX R  
CHAR/BYTE/STRING ROUTINES

TRUCST  
003

C NICE1 > NICE2

C

520 TRUCST=((KOP.EQ.'>').OR.(KOP.EQ.'>=').OR.(KOP.EQ.'<>'))

C

C

C DONE

C

900 RETURN  
END

**DAM PACKAGE APPENDIX 3  
SORT ROUTINES**

**PREFACE-3  
001**

**PREFACE TO APPENDIX 3  
-----**

**THIS APPENDIX CONTAINS SORT UTILITY ROUTINES FOR THE DAM PACKAGE.**

**DAN PACKAGE APPENDIX 8  
SORT ROUTINES**

**APPENDIX-8  
001**

SPRT.SC DAN.PREFACE-8  
SPRT.SC DAN.APPENDIX-8  
SPRT.SC DAN.ISRTBA  
SPRT.SC DAN.ISRTBD  
SPRT.SC DAN.ISRTBA  
SPRT.SC DAN.ISRTBD  
SPRT.SC DAN.ISRTSA  
SPRT.SC DAN.ISRTSD  
SPRT.SC DAN.TSRTMS

. (0009) SET TABS 8 12 6 31  
.  
. INTEGER BUBBLE SORT ASCENDING  
. INTEGER BUBBLE SORT DESCENDING  
. INTEGER HIBBARD'S SHELLSORT ASCENDING  
. INTEGER HIBBARD'S SHELLSORT DESCENDING  
. INTEGER SHUTTLE SORT ASCENDING  
. INTEGER SHUTTLE SORT DESCENDING  
. TAGSORT USING HIBBARD'S SHELLSORT

DAM PACKAGE APPENDIX 9  
SORT ROUTINES

ISRTBA  
001

```

      SUBROUTINE ISRTBA( 8 INTEGER BUBBLE SORT ASCENDING
      U IRAY.           8 ARRAY OF INTEGERS TO BE SORTED
      I NWORDS)        8 NUMBER OF WORDS IN ARRAY
      -----
C
C
C (E H SCHLOSSER)
C
C
C THIS SUBROUTINE SORTS AN ARRAY OF INTEGERS IN ASCENDING SEQUENCE USING A
C BUBBLESORT. MODIFIED SO IT IS LOGICALLY EQUIVALENT TO SHUTTLESORT. BUT
C REQUIRING SLIGHTLY LESS CORE. THIS SORT IS VERY EFFICIENT WHEN (AND ONLY
C WHEN) THE ARRAY IS ALREADY GROSSLY SORTED IN THE PROPER SEQUENCE.
C
      DIMENSION IRAY(1)
      IF(NWORDS.LT.2) GO TO 900
      N=2
      DO TO 200
      100 N=N+1
      150 IF(N.GT.NWORDS) GO TO 900
      200 IF(IRAY(N).GE. IRAY(N-1)) GO TO 100
      NTEMP=N
      300 ITEMP=IRAY(N)
      IRAY(N)=IRAY(N-1)
      IRAY(N-1)=ITEMP
      IF(N.LT.3) GO TO 400
      N=N-1
      IF(IRAY(N).LT. IRAY(N-1)) GO TO 300
      400 N=NTEMP+1
      DO TO 150
      900 RETURN
      END

```

**SAN PACKAGE APPENDIX 8  
SORT ROUTINES**

**ISRTSD  
001**

**SUBROUTINE ISRTSD:   8 INTEGER BUBBLE SORT DESCENDING  
U IRAY,               8 ARRAY OF INTEGERS TO BE SORTED  
I NWORDS)           8 NUMBER OF WORDS IN ARRAY**  
-----

**C  
C  
C (E H SCHLOSSER)  
C  
C THIS SUBROUTINE SORTS AN ARRAY OF INTEGERS IN DESCENDING SEQUENCE USING A  
C BUBBLESORT. MODIFIED SO IT IS LOGICALLY EQUIVALENT TO SHUTTLESORT. BUT  
C REQUIRING SLIGHTLY LESS CORE. THIS SORT IS VERY EFFICIENT WHEN (AND ONLY  
C WHEN) THE ARRAY IS ALREADY GROSSLY SORTED IN THE PROPER SEQUENCE.  
C**

**DIMENSION IRAY(1)  
IF(NWORDS.LT.2) GO TO 900  
N=2  
GO TO 200  
100 N=N+1  
150 IF(N.GT.NWORDS) GO TO 900  
200 IF(IRAY(N).LE.IRAY(N-1)) GO TO 100  
NTEMP=N  
300 ITEMP=IRAY(N)  
IRAY(N)=IRAY(N-1)  
IRAY(N-1)=ITEMP  
IF(N.LT.3) GO TO 400  
N=N-1  
IF(IRAY(N).GT.IRAY(N-1)) GO TO 300  
400 N=NTEMP+1  
GO TO 150  
900 RETURN  
END**

DAN PACKAGE APPENDIX 8  
SORT ROUTINES

ISRTMA  
001

SUBROUTINE ISRTMA( 8 INTEGER HIBBARD'S SHELLSORT ASCENDING  
U IRAY, 8 ARRAY OF INTEGERS I: TO SORT 0: SORTED  
I NWORDS) 8 NUMBER OF WORDS IN ARRAY  
-----

```

C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      07/10/73      ORIGINAL CODE
C      E M SCHLOSSER      LEMSCO  02/07/80      UPGRADE DOCUMENTATION
C
C METHOD
C -----
C
C      SORT INTEGERS IN ASCENDING SEQUENCE USING HIBBARD'S IMPROVED
C      SHELLSORT (CACH ALGORITHM + 201).
C
C      THIS SIMPLE TECHNIQUE, AN EXTENSION OF SHUTTLESORT, IS QUITE EFFICIENT
C      FOR SORTING ARRAYS OF RANDOM NUMBERS. IT IS NOT AS FAST AS SHUTTLESORT
C      WHEN THE ARRAY IS ALREADY SORTED.
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C EXTERNAL REFERENCES
C -----
C
C      NONE.
C
C EXCEPTIONS
C -----
C
C      NONE.
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER IRAY(1)      8 ARGUMENT
C      INTEGER KIIST        8 COMPARISON DISTANCE
C      INTEGER KS            8 COMPARISON SUBSCRIPT
C      INTEGER KSHAX        8 MAXIMUM COMPARISON SUBSCRIPT
C      INTEGER KSMI         8 CURRENT HIGHEST COMPARISON SUBSCRIPT

```



DAN PACKAGE APPENDIX 3  
 SORT ROUTINES

ISRTNA  
 002

```

      INTEGER INTMP      & TEMPORARY
C
C
C PROCEDURE
C -----
C
      IF(NWORDS.LT.2) GO TO 900
C
      KDIST=NWORDS
200    CONTINUE      & WHILE KDIST <> 1 DO
          KDIST=2+((KDIST+2)/4)-1
          KSMAX=NWORDS-KDIST
          DO 400 KSHI=1,KSMAX
              DO 300 KS=KSHI,1.-KDIST
                  IF(IRAY(KS).LE.IRAY(KS+KDIST)) GO TO 400
                  INTMP=IRAY(KS)
                  IRAY(KS)=IRAY(KS+KDIST)
                  IRAY(KS+KDIST)=INTMP
300          CONTINUE
400    CONTINUE
          IF(KDIST.NE.1) GO TO 200
C
900    RETURN
      END
  
```

187400  
001

\*\*\*\*\*

|   |               |                 |                       |
|---|---------------|-----------------|-----------------------|
| C |               |                 |                       |
| C | E H SCHLOSSER | LEC 07/10/73    | ORIGINAL CODE         |
| C | E H SCHLOSSER | LENSCO 02/07/80 | UPGRADE DOCUMENTATION |

```

C
C      SORT INTEGERS IN DESCENDING SEQUENCE USING HIBBARD'S IMPROVED
C      SHELLSORT (CACH ALGORITHM # 201).

```

C  
C  
C MACHINE-DEPENDENT CODE  
C

C  
C  
C **EXTERNAL REFERENCES**  
C

EXCEPTIONS

GLOBAL DECLARATIONS
\*\*\*\*\*

**LOCAL DECLARATIONS**

**8-7**

DAN PACKAGE APPENDIX 3  
 SORT ROUTINES

ISRTNO  
 662

```

      INTEGER INTMP      8 TEMPORARY
C
C
C PROCEDURE
C -----
C
      IF(NWORDS.LT.2) GO TO 900
C
      KDIST=NWORDS
200    CONTINUE      8 WHILE KDIST <> 1 DO
          KDIST=2*((KDIST+2)/4)-1
          KSMAX=NWORDS-KDIST
          DO 400 KSHI=1,KSMAX
              DO 300 KS=KSHI,1.-KDIST
                  IF(IRAY(KS).GE.(IRAY(KS+KDIST))) GO TO 400
                  INTMP=IRAY(KS)
                  IRAY(KS)=IRAY(KS+KDIST)
                  IRAY(KS+KDIST)=INTMP
300          CONTINUE
400          CONTINUE
          IF(KDIST.NE.1) GO TO 200
C
900    RETURN
      END
  
```

DAN PACKAGE APPENDIX 8  
SORT ROUTINES

ISRTSA  
001

SUBROUTINE ISRTSA( 8 INTEGER SHUTTLE SORT ASCENDING  
U IRAY, 8 ARRAY OF INTEGERS 1: TO SORT 0: SORTED  
I NHORDS) 8 NUMBER OF WORDS IN ARRAY  
-----

```

C
C
C
C HISTORY
C -----
C
C      E M SCHLOSSER      LEC      07/10/73      ORIGINAL CODE
C      E M SCHLOSSER      LEHSCO 02/07/80      UPGRADE DOCUMENTATION
C
C
C METHOD
C -----
C
C      SORT INTEGERS IN ASCENDING SEQUENCE USING SHUTTLESORT (CACH
C      ALGORITHM # 175). THIS TECHNIQUE IS VERY EFFICIENT WHEN
C      (AND ONLY WHEN) THE ARRAY IS ALREADY GROSSLY SORTED IN THE PROPER
C      SEQUENCE.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      NONE.
C
C
C EXCEPTIONS
C -----
C
C      NONE.
C
C
C GLOBAL DECLARATIONS
C -----
C
C      NONE.
C
C
C LOCAL DECLARATIONS
C -----
C
C      INTEGER IRAY(1)      8 ARGUMENT
C      INTEGER KS           8 COMPARISON SUBSCRIPT
C      INTEGER KSHAX        8 MAXIMUM COMPARISON SUBSCRIPT
C      INTEGER KSHI         8 CURRENT HIGHEST COMPARISON SUBSCRIPT
C      INTEGER ITEMP        8 TEMPORARY

```

DAM PACKAGE APPENDIX S  
SORT ROUTINES

ISRTSA  
682

C PROCEDURE

C -----

C

IF(NWORDS.LT.2) GO TO 900

C

KSMAX=NWORDS-1

DO 400 KSHI=1,KSMAX

DO 300 KS=KSHI,1,-1

IF(IRAY(KS).LE.IRAY(KS+1)) GOTO 400

INTMP=IRAY(KS)

IRAY(KS)=IRAY(KS+1)

IRAY(KS+1)=INTMP

300 CONTINUE

400 CONTINUE

C

900 RETURN

END

DAN PACKAGE APPENDIX 3  
SORT ROUTINES

ISRTSD  
001

SUBROUTINE ISRTSD( 3 INTEGER SHUTTLE SORT DESCENDING  
U IRAY, 3 ARRAY OF INTEGERS 1: TO SORT 0: SORTED  
I NMWORDS) 3 NUMBER OF WORDS IN ARRAY  
-----

C

C

C

C HISTORY

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

E H SCHLOSSER LEC 07/10/73 ORIGINAL CODE  
E H SCHLOSSER LEMSCO 02/07/80 UPGRADE DOCUMENTATION

C METHOD

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

INTEGER IRAY(1) 3 ARGUMENT  
INTEGER KS 3 COMPARISON SUBSCRIPT  
INTEGER KSMAX 3 MAXIMUM COMPARISON SUBSCRIPT  
INTEGER KSHI 3 CURRENT HIGHEST COMPARISON SUBSCRIPT  
INTEGER INTMP 3 TEMPORARY

C

C

DAM PACKAGE APPENDIX S  
SORT ROUTINES

ISRTSD  
002

C PROCEDURE

C -----

C

IF(NWORDS.LT.2) GO TO 900

C

KSMAX=NWORDS-1

DO 400 KSHI=1,KSMAX

DO 300 KS=KSHI,1,-1

IF(IRAY(KS).GE.IRAY(KS+1)) GOTO 400

INTMP=IRAY(KS)

IRAY(KS)=IRAY(KS+1)

IRAY(KS+1)=INTMP

300 CONTINUE

400 CONTINUE

C

900 RETURN

END

DAN PACKAGE APPENDIX 3  
SORT ROUTINES

TSRTHS  
001

```

SUBROUTINE TSRTHS( 3 TAGSORT USING HIBBARD'S SHELLSORT
0 INTAG. 3 INTEGER ARRAY OF TAGS (POINTERS) SORTED BY CHAR STRING KEY
.
1 KRECRO. 3 ARRAY OF FIXED LENGTH PACKED CHARACTER RECORDS TO SORT
1 NHIREC. 3 NUMBER OF WORDS IN 1 RECORD
1 NRECS. 3 NUMBER OF RECORDS (ALSO NUMBER OF TAGS)
1 KEYLOC. 3 CHARACTER LOCATION WITHIN RECORD WHERE SORT KEY BEGINS
1 KEYLEN. 3 LENGTH & DIRECTION OF SORT KEY (POSITIVE = L TO R)
1 KSORDR) 3 KEYSORT ORDER ('A' FOR ASCENDING, 'D' FOR DESCENDING)
          (CHAR LOCATION COUNTED FROM 1 AT LEFT OF STRING)
-----
C
C
C
C HISTORY
C -----
C
C      E H SCHLOSSER      LEHSCO 09/08/80      ORIGINAL CODE
C
C
C
C METHOD
C -----
C
C      CHECK FOR VALID ARGUMENTS.
C      SORT TAGS IN SPECIFIED KEY SEQUENCE USING HIBBARD'S IMPROVED
C      SHELLSORT (CACH ALGORITHM # 201).
C
C      THIS SIMPLE TECHNIQUE, AN EXTENSION OF SHUTTLESORT, IS QUITE EFFICIENT
C      FOR SORTING DATA IN RANDOM ORDER. IT IS NOT AS FAST AS SHUTTLESORT
C      WHEN THE DATA ARE ALREADY SORTED.
C
C
C MACHINE-DEPENDENT CODE
C -----
C
C      NONE.
C
C
C EXTERNAL REFERENCES
C -----
C
C      INTEGER NI4NC      3 NUMBER OF INTEGERS (WORDS) FOR NUMBER OF CHARACTERS
C      LOGICAL TRUCST      3 TRUTH VALUE OF CHARACTER STRING COMPARISON
C
C
C EXCEPTIONS
C -----
C
C      1. THE FOLLOWING INVALID ARGUMENT VALUES PREVENT SORTING AND CAUSE
C      INTAG(1) TO BE SET TO 0:
C          NHIREC < 1
C          NRECS < 1
C          SORT KEY WHOLLY OR PARTIALLY OUTSIDE RECORD
C          KSORDR NOT 'A' OR 'D'
C
C
C GLOBAL DECLARATIONS

```



DAN PACKAGE APPENDIX 3  
SORT ROUTINES

TSRTHS  
002

```

C -----
C
C      NONE.
C
C LOCAL DECLARATIONS
C -----
C
      INTEGER INTAG(NRECS)      & INTEGER ARGUMENT
      INTEGER KRECRD(NWIREC,NRECS) & PACKED CHARACTER ARRAY ARGUMENT

      INTEGER KEYDIR      & DIRECTION OF SORT KEY (-1 OR +1)
      INTEGER KOP         & CHAR STRING COMPARISON OPERATOR '<' OR '>'
      INTEGER KDIST       & COMPARISON DISTANCE

      INTEGER KRN         & PRIMARY COMPARISON RECORD NUMBER
      INTEGER KRNDIS      & DISTANT COMPARISON RECORD NUMBER

      INTEGER KTN         & TAG NUMBER OF PRIMARY COMPARISON RECORD
      INTEGER KTNMAX      & MAXIMUM TAG NUMBER OF PCR
      INTEGER KTNHI       & CURRENT HIGHEST TAG NUMBER OF PCR

C
C
C PROCEDURE
C -----
C
C CHECK ARGUMENTS
C
      INTAG(1)=0      & FLAG FOR BAD ARGUMENT(S)

C
      KOP=' '
      IF(KSORDR.EQ.'A') KOP='<'
      IF(KSORDR.EQ.'D') KOP='>'
      IF(KOP.EQ.' ') GO TO 900

C
      IF(NWIREC.LT.1) GO TO 900
      IF(NRECS.LT.1) GO TO 900

C
      IF(KEYLOC.LT.1) GO TO 900
      IF(NI4NC(KEYLOC).GT.NWIREC) GO TO 900
      KEYDIR=MIN(1,MAX(-1,KEYLEN))
      IF(KEYLOC+KEYLEN-KEYDIR.LT.1) GO TO 900
      IF(NI4NC(KEYLOC+KEYLEN-KEYDIR).GT.NWIREC) GO TO 900

C
C
C INITIALIZE TAGS
C
      DO 100 KRN=1,NRECS
        INTAG(KRN)=KRN
      100 CONTINUE

C
C
C PERFORM SORT
C
      KDIST=NRECS

```

DAM PACKAGE APPENDIX S  
SORT ROUTINES

TSRTMS  
003

```

200      IF(KDIST.EQ.1) GO TO 900      & WHILE KDIST<>1 DO
      KDIST=2*((KDIST+2)/4)-1
      KTNMAX=NRECS-KDIST
      DO 400 KTNHI=1,KTNMAX
      DO 300 KTN=KTNHI,1,-KDIST
      KRN=INTAO(KTN)
      KRNOIS=INTAO(KTN-KDIST)
      IF(TRUCT(
      &      KRECRD(1,KRN),KEYLOC,KEYLEN,KOP,
      &      KRECRD(1,KRNOIS),KEYLOC,KEYLEN))
      &      GO TO 400
      INTAO(KTN)=KRNOIS
      INTAO(KTN-KDIST)=KRN
300      CONTINUE
400      CONTINUE
      GO TO 200
C
C
C DONE
C
900 RETURN
      END

```

ORIGINAL PAGE 1  
OF POOR QUALITY